VOLUME 05 | ISSUE 02

digit FastTrack

Fast track to OpenSolaris

Discover the geek in you. Participate in the **OpenSolaris Geek Hunt Contest** @ **http://www.thinkdigit.com/opensolaris** and win an **Acer Aspire Netbook**

- What is OpenSolaris
- OpenSolaris Installation
- Desktop and Applications
- Deep Dive in to ZFS Filesystem
- Dynamic Tracing with Dtrace
- Contribute to OpenSolaris with Source Juicer
- Careers in Open Source
- 5 step guide to contribute to Open Source
- What will be new in OpenSolaris 2010.03

A 9.9 Media Publication

digit
YOUR TECHNOLOGY NAVIGATOR
thinkdigit.com

Fast Track to Open Solaris

Credits

The People Behind This Book

Editorial

Editor	Robert Sovereign-Smith
Head-Copy Desk	Nash David
Writer	Kumar Abhishek, Leader – Mumbai OpenSolaris User Group

Design and Layout

Lead Designer	Vijay Padaya
Senior Designer	Baiju NV
Cover Design	Prashanth TR

February 2010
Free with Digit. Not to be sold separately. If you have paid separately for this book, please email the editor at editor@thinkdigit.com along with details of location of purchase, for appropriate action.

Contents

What you will learn...

We have two things to share with you. First, we want to give you a neat and simple guide to the OpenSolaris operating system. Second, through OpenSolaris, we want to show you how open source is not just about *using* open source software, but also contributing too. In a unique attempt, the Digit editorial team accepted contribution from members of the Mumbai OpenSolaris User Group for this *Fast Track to OpenSolaris*.

You can use this issue of Fast Track as a quick and dirty guide to OpenSolaris. It is not only for those who want to become open source developers, but also for those who're looking at options other than proprietary operating. Here's what to find:

Chapter 1: We introduce you to OpenSolaris by giving you a brief history and some of its features, along with what's different when compared to other popular operating systems.

Chapter 2: Here is discuss how to use the LiveCD and how to install OpenSolaris.

Chapter 3: In this chapter we tell you how to use OpenSolaris. How to install new applications and how to manage and update your system. We also describe how to troubleshoot your system and install device drivers. By end of this chapter you will be proficient in use of OpenSolaris.

Chapter 4: In this chapter we introduce one of the most distinct feature of OpenSolaris in detail, which is ZFS filesystem. This chapter will take you under the hood and explain you things in deeper detail than what an average user is expected to know.

If you're afraid of running commands on the terminal thinking something may go wrong then do not worry. After you finish installing OpenSolaris, just open a terminal and execute the command `$ pfexec beadm create backup-boot`. This will create a new boot environment for OpenSolaris and if you do manage to make something go wrong somehow, you can select this from the boot menu and unless you have done something very nasty with your hardware, your system should boot up as usual.

Chapter 5: In this chapter we cover a powerful developer tool offered in OpenSolaris called DTrace. It again takes you a level higher than an average user.

Chapter 6: Here we introduce you to world of contributing to open source. This chapter gives a step-by-step guide on how to port your own applications or other existing open source applications on OpenSolaris platform. This is a very easy exercise and will be a good point to start for those who want to try hands at becoming open source developers.

Chapter 7: In this chapter we cover an important topic of careers. We give you some tips and tricks and resources on how you can make a career out of OpenSolaris in specific and open source in general.

Chapter 8: This chapter covers all important resource you may need in order to use OpenSolaris or become a developer. We also present a five step guide on how to take up projects in OpenSolaris community and contribute code or documentations. At the end, we have prepared a checklist of things you should do after reading this Fast Track

Chapter 9: This chapter unveils some of the new features which will be available in OpenSolaris 2010.03 which will be released in March this year.

We didn't just want to introduce you to a cutting edge open source software, but also teach you how you could be a part of such a project and contribute. India, which is known all across the globe as a country which produces superior software talent, hardly finds open source contributors.

With thousands of engineering and technical colleges producing IT minds, it's possible to empower our country beyond imagination, we Indians collectively took up community oriented open source development to come up with technological solutions to our social problems such as fighting crime and better health care. But we understand that most of us do not know where to start and how to proceed in open source world. Through this *Fast Track*, we have taken a small step in educating our readers on how a typical open source community works in plane simple language by taking OpenSolaris community as an example. Should you have any questions feel free to discuss them at **www.thinkdigit.com/forum**. Now go ahead and enjoy OpenSolaris!

1 Getting started

OpenSolaris is the binary distribution of a mature, free and open source operating system, based around Sun Microsystems' Solaris operating system. The history of OpenSolaris begins in 1983 (27 years ago!) when Sun released its own flavour of Unix named SunOS 1.0. In June 1992, SunOS was renamed Solaris. With millions of installations all around the globe, Solaris is one of the most popular flavours of Unix in the world and is known for its scalability, security and stability. In 2005, Sun decided to open source the code base of Solaris and as the result of it, OpenSolaris was born. OpenSolaris is available for the x86 platform as well as the SPARC platform. Both 32-bit and 64-bit versions are available and can be downloaded from http://www.opensolaris.com.

1.1 Getting familiar

Like every other popular operating system, OpenSolaris has a GUI, which is based on GNOME. Thousands of applications, including the most popular open source applications Firefox, OpenOffice and Thunderbird have been ported to OpenSolaris. OpenSolaris is distributed under the Common Development and Distribution License (CDDL), which is a variant of the Mozilla licence used by Firefox. Because many applications, including the GNOME desktop environment, are common with various Linux distributions, you can confuse OpenSolaris to be yet another Linux distribution. However, OpenSolaris is not Linux! There are many unique features available on OpenSolaris that sets it apart from all the other operating systems such as Linux and Windows.

The term *OpenSolaris* can also have other meanings to depending on the context it is used. *OpenSolaris* is used to refer to:

- An operating system code base that is originally based on Sun's proprietary Solaris operating system.

Desktop of a freshly installed OpenSolaris system

- An open source development project.
- The community of users and developers who contribute to the project OpenSolaris.
- A free binary distribution.

Throughout this book, we'll be referring to the last meaning in the list, the term OpenSolaris means a free binary distribution of the OpenSolaris software that is available for redistribution under the terms of the Common Development and Distribution License (CDDL), unless otherwise stated.

1.2 Differences with Linux and Windows

Even though OpenSolaris is free and open source similar to Linux, it is different in many ways. Unlike the Linux distributions such as Ubuntu, Fedora and OpenSUSE, which are based on the Linux kernel, OpenSolaris is based on the Solaris kernel. This makes OpenSolaris fundamentally different from all the Linux based operating systems. But both OpenSolaris and Linux (and many other Unix like operating systems) are POSIX compliant (which is an industry standard for Unix based OS) and so thousands of applications which run on Linux based operating systems, also run on OpenSolaris.

Moreover, OpenSolaris has many unique features, which are not available on any flavour of Linux. For example, while Linux uses Ext3 or Ext4 file systems, OpenSolaris uses the ZFS file system which is a much more advanced file system. We will discuss ZFS in greater detail later in the book. OpenSolaris also provides developer tools like DTrace, which allow programmers and system administrators to probe the inner functioning of an applications while it is still running. DTrace will be covered in details as well.

Differences between Linux and OpenSolaris exists in licensing as well. While Linux is released under the GNU Public License (GPL), OpenSolaris is released under Common Development and Distribution License (CDDL). CDDL is based on Mozilla Public License (MPL) and unlike GPL, which is a project based license, CDDL is a file based license. This allows users and developers a greater level of freedom and permits both closed source as well as open source components to co-exist in the same project.

OpenSolaris is very different from the operating systems of the Microsoft Windows family. As stated earlier, unlike Windows, OpenSolaris is free and open source. Even though it is possible to run some of the applications developed for Windows using WINE, OpenSolaris and Windows are different in most of the technological aspects.

1.3 Key features

OpenSolaris provides many unmatched features in almost every aspect of its use.

For a Desktop User: If you want to use OpenSolaris as a desktop operating system, here are some features which would make many things simple and possible.

Image Packaging System (IPS): Finding, installing and managing applications has never been easier. The Image Packaging System allows users to search applications and safely install them without hassles from various software repositories online. It is also possible to set up local software repositories over a standalone machine or over any local area network (LAN) infrastructure.

Time Slider: Timeslider is a unique feature of OpenSolaris. By utilizing the capabilities of the underlying ZFS file system, Timeslider automatically keeps taking regular snapshots of your files and allows you to access files which you might have deleted or changed just by click of a button.

Device Driver Utility: OpenSolaris Device Driver Utility scans all the hardware and peripheral devices attached to the system and alerts the

user if device driver for any particular device is not available. It also allows user to submit the details of such device to the OpenSolaris Device Driver support group.

Popular software like Firefox, Thunderbird mail client and the OpenOffice productivity suite are available on the OpenSolaris platform. Funky 3D desktop effects are also possible through the Compiz Visual Effects which comes preloaded with OpenSolaris.

For Developers: The real fun of OpenSolaris is when you use it for development purposes. It provides a very unique set of tools which allow programmers to write better and more optimized programs and debug them easily. It also makes developing and deploying applications easy. Some of the cool features which would appeal developers are as follows.

DTrace: A unique feature of OpenSolaris, DTrace allows you to enable probes on a running application and get to know its inner workings and functioning. There are more than 60,000 probes which can be used to profile an application and new ones get added regularly. We will cover DTrace in detail later in the book. You can also refer to the DTrace Quick Start Guide which is available on the Digit DVD.

Service Management Facility (SMF): SMF is a very easy way of managing various services which run in the background to perform particular tasks. You can enable or disable any service with simple commands and easily deploy your own services. You can learn more on SMF in the Quick Start Guide available at http://www.sun.com/bigadmin/content/selfheal/smf-quickstart.jsp.

Network Virtualisation with Crossbow: You might have heard about creating virtual machines to run operating systems, but OpenSolaris has a unique feature called crossbow, which allows to virtualise the network. You can create multiple virtual Network Interface Cards (NIC), with their own IP and MAC addresses, based on a single NIC of your laptop or desktop and convert your laptop into a server like machine! You can learn more about Crossbow on http://hub.opensolaris.org/bin/view/Project+crossbow/.

ZFS File system: The ZFS file system is a revolutionary new file system that fundamentally changes the way file systems are administered, with features and benefits not found in any other file system available today. ZFS has been designed to be robust, scalable, and simple to administer. It is currently said to be the best file system in the world.

Note

There are many interesting things that can be done with Crossbow. Checkout our special section on Crossbow on http://www.thinkdigit.com/opensolaris.

Apache MySQL PHP (AMP) Stack: OpenSolaris allows you to install a precompiled and configured AMP development cluster for web application development using PHP and MySQL. You can start your PHP MySQL development in just one click and the best part of AMP on OpenSolaris is that you can use DTrace to debug and profile your PHP applications.

With great developer tools like Netbeans, Sun Studio and Sun Compliers, development becomes very easy on OpenSolaris. Be it web applications using PHP, system programming using C or Desktop applications using Java, OpenSolaris provides impressive tools to the developers.

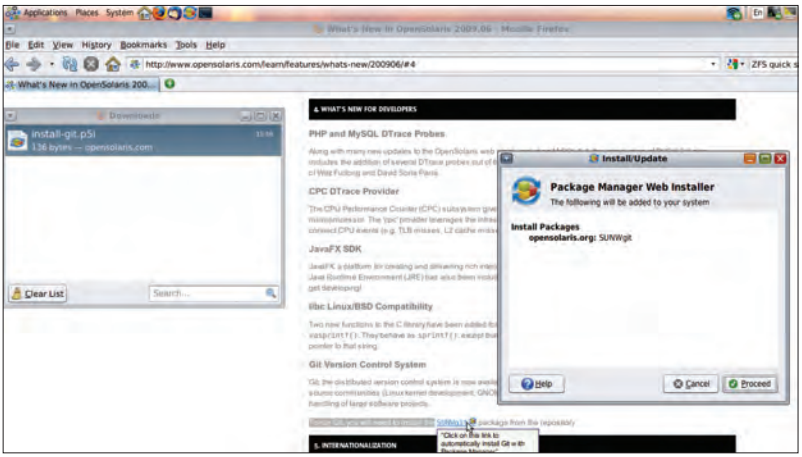
There are many more key features of OpenSolaris and you can learn more about it in the Getting Started with OpenSolaris 2009.06 document which is free to download from http://dlc.sun.com/osol/docs/downloads/minibook200906/en/820-7799-12-doc.pdf and is also available on the Digit DVD.

1.4 What's new in 2010.03?

The first release of OpenSolaris came out in May 2008 and was released as OpenSolaris 2008.05. The most recent release is OpenSolaris 2009.06 which was released in June 2009. The next release of OpenSolaris is scheduled to come out next month in March 2010. Many improvements were introduced in OpenSolaris 2009.06 which enhanced its usability and features. Some of the interesting new features of OpenSolaris 2009.06 are:

Multimedia With Codeina And Elisa: Codeina is a utility to allow users to install additional media plug-ins. GStreamer-based media applications will auto detect when users try to play a media file for which there is a plug-in available from the on-line Fluendo store, and will step the user through the process of downloading and installing the plug-in. Some plug-ins are free and some are available for a fee. For the first time on OpenSolaris, Elisa, the free and open source media centre is now available, connecting the internet to an all-in-one media player. Watch your photos with previews and nicely animated slide shows. Browse the internet, with everything from Flickr to YouTube and other popular internet services.

Package Manager: The Package Manager has received a number of improvements particularly around start-up performance and the user experience of the application. Package Manager now has a new start-up page, along with improvements to the search functionality that allows search across multiple repositories. Additionally, OpenSolaris 2009.06 brings a new MIME association (.p5i) to allow single click installs while browsing the web, which is quite unique to OpenSolaris.

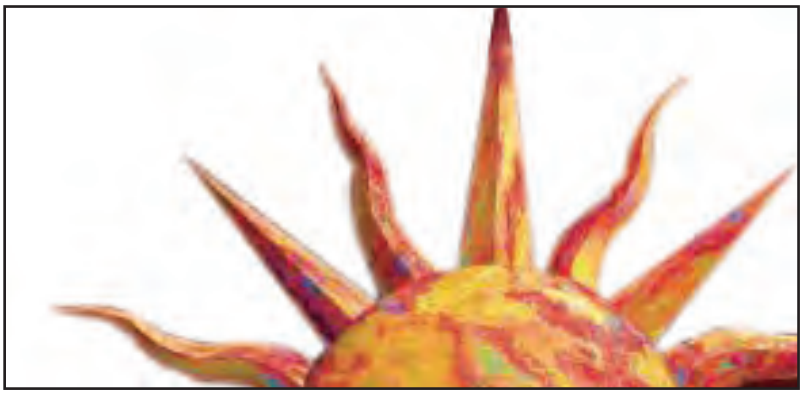


You can install or update packages right out of FireFox using Package Manager Web Installer!

Greater Windows interoperability with OpenSolaris CIFS: OpenSolaris CIFS (Common Internet File System) service now includes many new features such as host-based access control which allows a CIFS server to restrict access to specific clients by IP address, ACLs (access control lists) on shares, and client-side caching of offline files and synchronization when reconnected.

There are many more innovative features that has been introduced into OpenSolaris 2009.06. Visit http://www.opensolaris.com/learn/features/whats-new/200906/ to learn more on what else in new in OpenSolaris 2009.06.

The next version of OpenSolaris will be available in March 2010 and apart from an updated GNOME (GNOME 2.28), it will have many new feature updates in IPS, ZFS and Networking. Check out www.thinkdigit.com/opensolaris for more on OpenSolaris 2010.03.

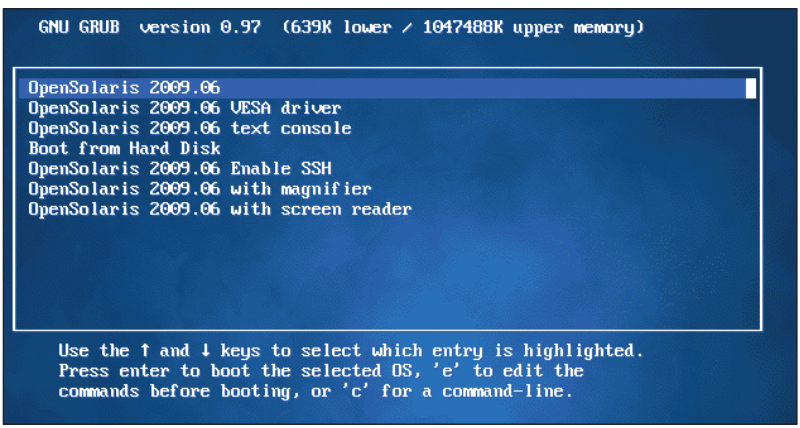


2 Installation

OpenSolaris is a LiveCD distribution, and so, you can try out OpenSolaris without actually installing it on your hard disk. Once you decide to install it on your physical drive, or inside a virtual machine, you can use the same LiveCD to do so, all with the click of a mouse. In this section, we will describe how to use the LiveCD and how to install OpenSolaris on to your computer.

2.1 Using the LiveCD

The LiveCD can be used by booting your computer through it. Put the LiveCD into your machine's CD drive and reboot. Once the machine



OpenSolaris 2009.06 LiveCD desktop with Device Driver Utility

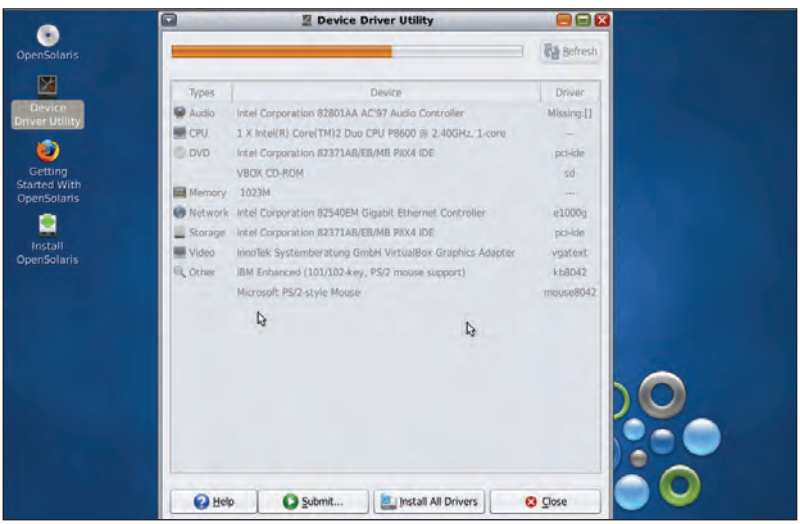
```
22. Latin-American
To select the keyboard layout, enter a number [default 43]:

 1. Arabic
 2. Chinese - Simplified
 3. Chinese - Traditional
 4. Czech
 5. Dutch
 6. English
 7. French
 8. German
 9. Greek
10. Hebrew
11. Hungarian
12. Indonesian
13. Italian
14. Japanese
15. Korean
16. Polish
17. Portuguese - Brazil
18. Russian
19. Slovak
20. Spanish
```

The keybord layout and language selection menu of OpenSolaris 2009.06 Live CD

boots from this CD you'll see a GRUB screen.

In the GRUB screen, select the first boot option (OpenSolaris 2009.06). The system will start booting from the LiveCD and soon you'd be asked to select the keyboard layout and language. In order to boot from the CD you might need to change the boot order of your machine by modifying



OpenSolaris 2009.06 LiveCD desktop with Device Driver Utility

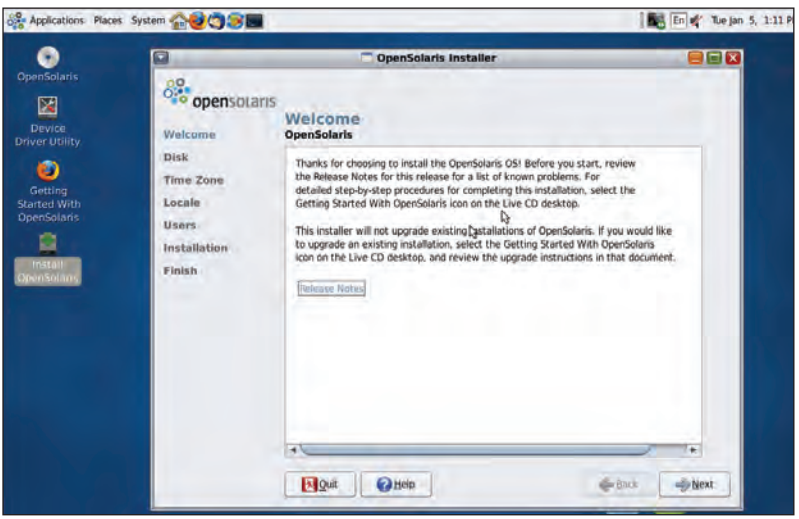
the BIOS settings. Please make sure your BIOS is configured to boot from CD before it boots from the hard disk.

When prompted, press `[Enter]` to accept the default values for keyboard (en_US) and language (English) and you would soon see the OpenSolaris desktop. After selecting keyboard layout and language, you will get a prompt which says console login. Please do not enter anything here. Wait for the GUI, to load automatically.

Once the GNOME desktop is running, you will see the three icons on the desktop – Device Driver Utility, Getting Started With OpenSolaris, and Install OpenSolaris. You can use Device Driver Utility to scan your computer and see if all the device drivers are present or drivers for some devices are missing. Unless the drivers for some critical devices, like the graphics adapter are missing, you can go ahead with the installation.

2.2 Partitioning the disk

Once you have explored the LiveCD and have made up your mind to install OpenSolaris, you will need to prepare your computer for it. The first step is to prepare a partition in your disk. OpenSolaris needs a primary partition for installation and the minimum recommended size is 10 GB.

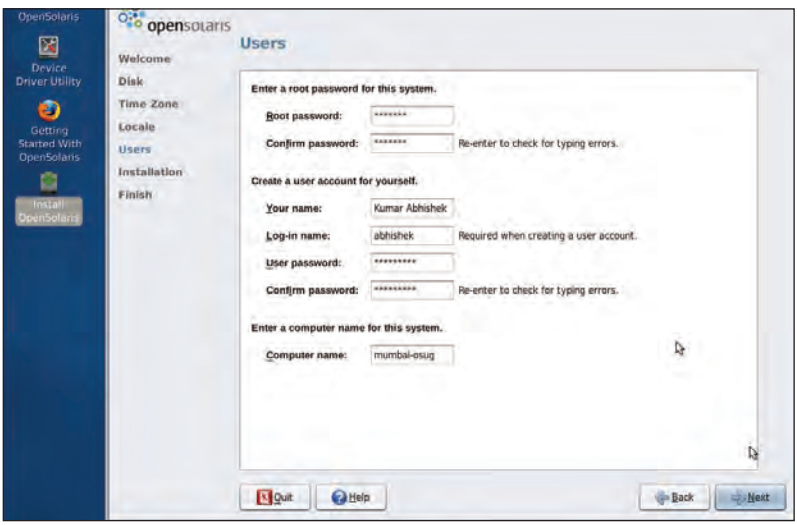


Welcome Screen of OpenSolaris 2009.06 Installer

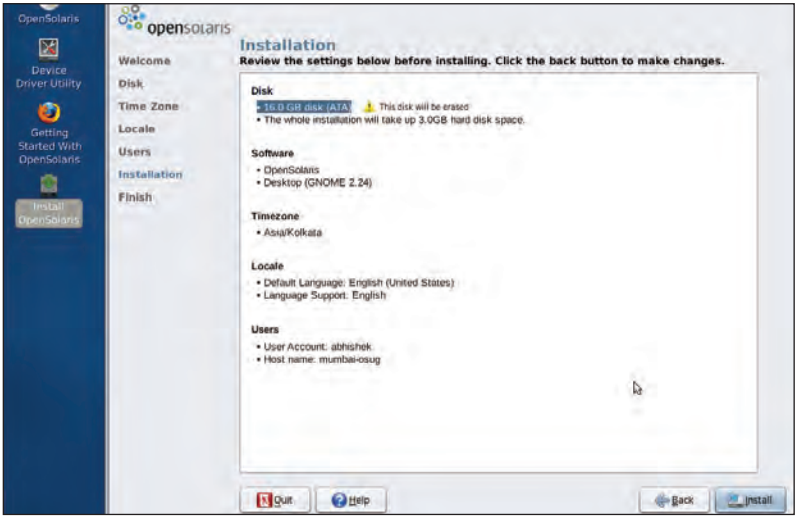
Select the Partition On Which You Would Like to Install OpenSolaris 2009.06

2.3 Installation and setting up the boot loader

Once you've created the necessary primary partition, it's time to boot



Selecting root password, and creating the first user account



Selecting root password, and creating the first user account

from the LiveCD and start the installation. Once the system boots up, click on the Install OpenSolaris icon and follow the on screen instruction

OpenSolaris installer summary screen

to complete the installation.

OpenSolaris requires a minimum of 512 MB RAM. We recommend 1 GB RAM is recommended to use developer tools such as Netbeans and Sun Studio. If installing OpenSolaris in VirtualBox, we recommend 1.5 GB RAM.

Click `Next`. You'll see the screen to select the partition. On this screen, select the partition you'd like to install OpenSolaris. If you want to use the whole disk to install OpenSolaris (which you would want to do if you're installing OpenSolaris on a virtual hard disk using VirtualBox, or if you want to use OpenSolaris as your only operating system.

Gparted

You can use Gparted (which is bundled in the DVD) to create a new primary partition. Check out http://www.wikihow.com/Use-Gparted if you want to learn how to use Gparted. You can learn more on how to set up the partition on http://dlc.sun.com/osol/docs/content/2009.06/getstart/partition.html]. The partition type should be set to linux-swap in order to be recognized by the installer.

Next comes your timezone, locale (language) and setting up user accounts. Once you click Next, your system is ready to install OpenSolaris. A confirmation screen would be presented where the summary of the configuration is mentioned. Click `Next` to start installation. OpenSolaris Has a GUI based installer which makes the whole installation process very simple and interactive. If you have Linux installed on your machine, then the Linux GRUB will get overwritten. You will have to modify the menu.lst file of OpenSolaris GRUB to configure Linux and OpenSolaris in dual boot. Please read the instructions on how to edit the GRUB menu at http://tinyurl.com/editgrub.

If you have Windows installed on your computer, the OpenSolaris Boot Loader (which is a modified version of GRUB) will automatically identify it and set it up in the Boot Menu. Therefore, it is not difficult to set up OpenSolaris as dual boot with Windows. If Linux is your other operating system, make a copy of the Linux grub menu, as you'll need to add its entries to OpenSolaris' GRUB menu.

The installation typically takes between 15 to 25 minutes depending on your system speed. Once the installation is complete, the installer will ask for a reboot. Eject the LiveCD, and click on Reboot to complete the installation. It's always a good idea to register your copy of OpenSolaris. It will allow you to keep yourself updated and receive the latest information by email. Registration is free and you can do that by clicking the Register OpenSolaris icon after finishing

the installation.

2.4 Upgrading from previous versions of OpenSolaris

If you have a previous version of OpenSolaris installed on your system, then upgrading to a new version is very simple. You can do it in two ways. First, open the Update Manager. To do this, go to `System > Administration > Update Manager`. Click on `Update All`. It will prompt you to create a new Boot Environment and your system will get updated. This is one of the special features of OpenSolaris. Once you update your system, a new boot environment is created and added to the GRUB menu. Using the cloning feature of underlying ZFS file system, your earlier version is retained. You can boot into your old installation by selecting the appropriate menu entry from GRUB during boot.

Quick Tip

If you do not want to modify your hard disk and still want to play with OpenSolaris, you can do so by installing OpenSolaris 2009.06 inside VirtualBox. Install VirtualBox, which is bundled in the DVD, and follow the instructions in http://dlc.sun.com/osol/docs/content/dev/getstart/virtualbox.html or in the OpenSolaris 2009.06 Getting Started Guide which is also bundled in the DVD, to carry out the installation. You can also check out the Video Tutorial section of Fast Track Companion site http://www.thinkdigit.com/opensolaris to learn how to Install OpenSolaris in VirtualBox

If you do not want your old installation to remain on this disk then boot into the new (updated) boot environment and use the Manage Boot Environment menu of the Package Manager to delete the old BE. Clones are not copies of file systems, but only a reference to it and so they do not occupy extra storage. Only the changes made in the clones are saved on the hard disk.

If you're a terminal guy, execute `pfexec pkg image-update`. Your system will get updated to the latest release. For this you'd need an internet connection. Check out the videos on the Fast Track Companion site http://www.thinkdigit.com/opensolaris to learn more on how to update OpenSolaris to the latest release. A new build of OpenSolaris comes up roughly every 2 weeks or so. You you want to get the feel of the bleeding edge of OpenSolaris then you can install these builds using the package manager. You will have to add the http://pkg.opensolaris.org/dev repository using the Manage Repositories menu of Package Manager. And then run the command `pfexec pkg image-update` and you can get your hands on the latest builds of OpenSolaris. Check out http://pkg.opensolaris.org/dev for more on this.

2.5 Automated installer

Imagine a situation where you have to install OpenSolaris with same settings in not one, but ten different machine at the same time. As OpenSolaris is very often used on servers and network of computers, it would be very time consuming to manually install OpenSolaris on every terminal. Therefore, OpenSolaris comes up with a network based Automated Installer feature as well. Yo can lean more about Automated Installer at http://tinyurl.com/autoinst.

2.6 Setting up user accounts and groups

After installing OpenSolaris, you're now ready to create new user account. This is a very simple process. Just click `System > Administration > User and Groups`.

One interesting thing about users in OpenSolaris is its Role Based Access Control (RBAC). Using RBAC it is possible to assign a user only enough rights that he or she may need in order to get their job done. This means that one need not have root access if their work only involves performing only a few task which requires root access. RBAC can be configured in order to allow them only certain rights, thereby making the system more secure. You can learn more about RBAC in the

Creating new users and groups and editing the existing users is very easy in OpenSolaris

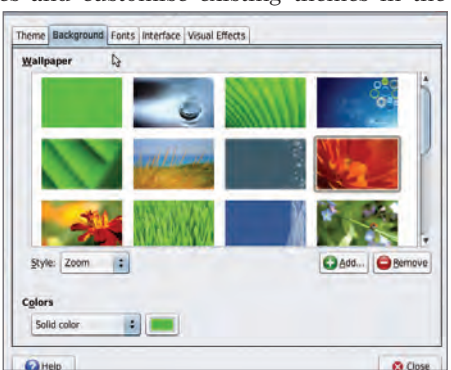
document *Getting Started with OpenSolaris 2009.06 Guide*, bundled in the DVD or available at http://dlc.sun.com/osol/docs/content/dev/getstart/.

3 Desktop and applications

The user interface of OpenSolaris is based on the GNOME desktop environment. So if you have ever used a GNOME-based Linux distribution (like Ubuntu or Fedora) you may find many similarities between the look and feel. But GNOME in OpenSolaris has been modified to accommodate many unique features.

3.1 Customising the user interface

OpenSolaris goes a long way in letting you customise your desktop. You can customise almost every aspect of your GNOME desktop like its theme and wallpaper and fonts. Just click on any blank area of desktop and select Desktop Appearance menu to access the Appearance Preference window. Here, you can apply new themes and customise existing themes in the Themes tab. You can change the background using the Background tab, and modify the fonts from the Fonts tab. You can also control certain aspects of the interface by accessing the interface tab and you can enable, disable or configure the 3D effects from the Visual Effects tab.



The OpenSolaris Desktop Appearance Preference Window

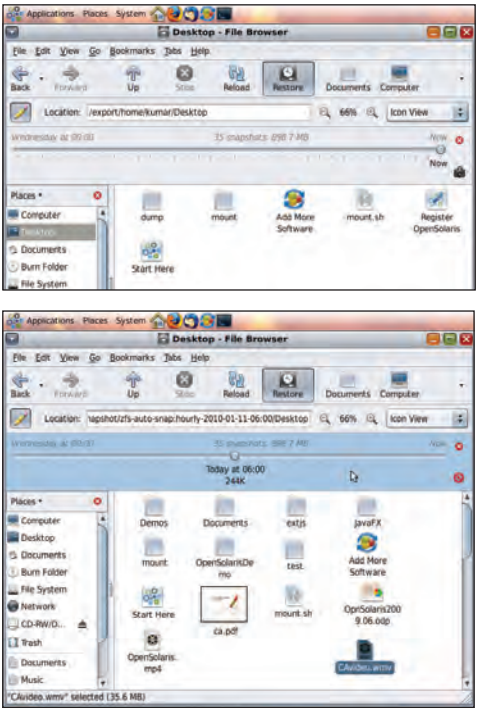
3.1.1 BeleniX distribution

Even though K Desktop Environment (KDE) can be build and installed on OpenSolaris the easiest

way to get KDE on top of OpenSolaris kernel is BeleniX. BeleniX is a distribution of OpenSolaris which uses KDE as its desktop environment. BeleniX can be easily installed along with OpenSolaris by creating a new boot environment (which is done automatically during installing BeleniX by following the steps mentioned below). The latest release of BeleniX is 0.8 Beta 1 and it can be installed via the BeleniX Network Installer. After you finish installing OpenSolaris 2009.06, connect your computer to the internet, open a terminal and follow these steps if you want to install BeleniX:

```
$ wget http://www.belenix.org/binfiles/install_belenix
$ chmod +x ./install_belenix
$ pfexec ./install_belenix
```

You can run `./install_belenix -help` to see a list of the options.



Using Time slider to browse files that have been deleted or changed

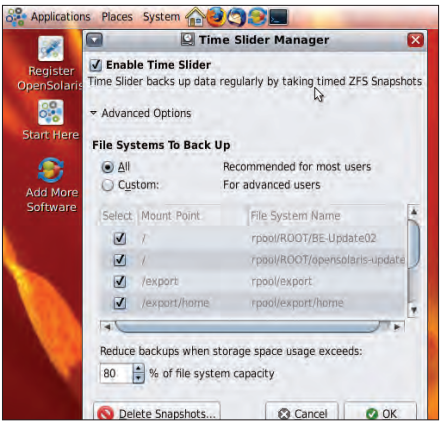
It will take some time to download all the binaries of BeleniX and after the installation is complete, it will be installed in a new Boot Environment, without modifying anything in your OpenSolaris installation. You can learn more about BeleniX on http://belenix.org.

3.2 Internet in OpenSolaris

Configuring a network in OpenSolaris is simple. It comes with Network Automagic which automatically connects your system with your ISP or LAN if your ISP or LAN follows DHCP, which is the case with most broadband internet service providers such as BSNL and Reliance. If you want to

manually configure the network then go to `System > Administration > Network`. Wi-Fi networks are also automatically configured by Network Automagic. To learn more about Network Automagic (NWAM), go to http://tr.im/nwamd.

3.3 Time Slider



Time Slider Manager

Time slider is the one of the best and most unique feature of OpenSolaris interface. Usually if you change or delete a file there are no easy way to get it back. Very often you might have multiple copies of same set of files with little changes in them. If enabled, Time slider automatically keeps taking a snapshot of your folders and files which you can easily revert back and forth! Take a manual snapshot with a click of a mouse, and revert to it at a later time. Time Slider is now easier than ever to use, with improved file manager integration with the ability of viewing your snapshots for particular files and folders.

You can also compare different versions of a image and plain text files using this feature.

Time slider is not enabled by default and you will have to manually enable it from the Time Slider Manager which you can access from `System > Administration > Time Slider`.

3.4 Software resources for OpenSolaris

There are thousands of applications available on OpenSolaris platform. Most of the famous open source application run on OpenSolaris. But what makes OpenSolaris a usable operating system is its ease of use when it comes to extending your operating system by installing new applications. In this section we cover how to use IPS to install, update and manage applications on OpenSolaris.

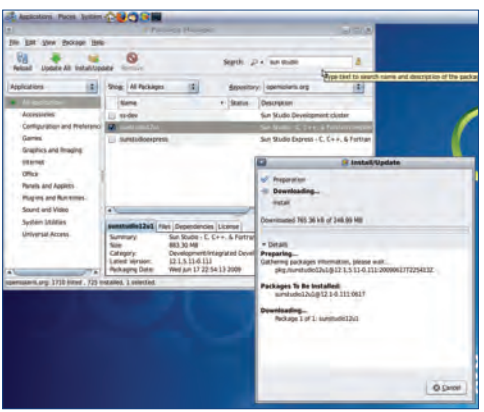
3.4.1 Using IPS to upgrade and install new applications

OpenSolaris since its first release, came up with a new Image Packaging System (IPS) for installing and managing applications. IPS provides a very simple interface to obtain new packages and update or remove existing packages. It can be accessed using both a GUI, and command line and also through the Firefox web browser!

Package management in Solaris is based on SVR4 style packges, which can be managed using using utilities such as `pkgadd` and `pkgrm`. But it does not have a GUI and if an application has dependencies on other applications (it requires you to have some other application installed on your system before installing it) then resolving these dependencies can some time be difficult. IPS has many advantages, both for users as well as developers, which we will cover in this section. We will also show some examples of how to install SVR4 style packages using `pkgadd` utility.

All thee user interface of Image Packaging Systems (IPS)

In order to install a new application using IPS, open the Package Manager by either clicking on Add More Software icon or clicking on `System > Administration > Package Manager`. Now you can use this GUI to search for packages and install them. The same GUI can be used to update or remove the packages as well.



Installing Sun Studio using Package Manager

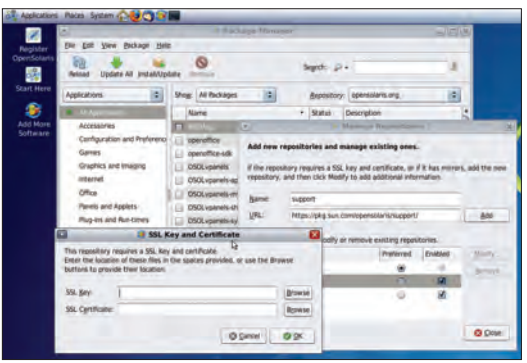
Just search for a package

using the search box. You can use the package name, or any keyword which might match the description of the package you want to install. A list of all the packages which match the keyword would appear as a list and you can then select the check box and click on the Install/Update button to install the application. If the Status column of the package has a check mark, it means that the package is already installed. In order to uninstall a package, click on the check box and then on Remove.

Package Manager allows multiple software repositories to be configured at the same time. So you can search and download packages from many places over the network and that too, in the click of a button. Read on to the next section on how to add repositories into your Package Manager.

3.4.2 Add new repositories to Package Manager

IPS brings a very useful feature to OpenSolaris, which allows users to easily find and download packages along with their dependencies from secure and trusted location and install them. This is done through repositories which can be accessed using IPS to download and install applications.



Adding the OpenSolaris Support Repository, which requires a SSL Key and Certificate

There are many software repositories, some of which are supported by Sun Microsystems Inc. and others are community supported. By default, the /release (http://pkg.opensolaris.org/release) repository is preconfigured with the Package Manager. As per your need you can add new repositories by following the simple steps.

Open the Package Manager (`System > Administration > Package Manager`) and click on `File > Manage Repositories`. This will bring up the Manage Repositories Window. Now you will need to give a name to the repository you are going to add. It can be any name, but it should not contain any invalid characters. Then you will need to provide the address

of the IPS repository which is the link to the repository. Click Add in order to add the repository. Some repositories might need a SSL certificate (as is the case with some special repositories, which are available only to those customers who buy support or carry software which cannot be freely redistributed). The Package Manager will prompt you to provide the SSL key and SSL certificate files when you click the add button.

Quick Tip

You can download hundreds of themes, wallpapers and splash screens for free from http://gnome-look.org/ Those who are die hard terminal fans, can disable the GNOME desktop environment and configure a terminal-only system by running: $ pfexec svcadm disable gdm. If you want to get the GNOME GUI back, then simple enable the gdm service by running $ pfexec svcadm enable gdm.

Once the repository is added, you can select the newly added repository from the drop down list and search and install packages from it. Note that you can search packages from only one repository at a time, so make sure that you have selected the correct repository in the Repository drop down list.

3.4.3 Free IPS repositories

Some of the popular community supported IPS repositories which you can use to download applications for free:

- http://pkg.opensolaris.org/contrib/

In this repository, the community contributed applications are available for download for free.

- http://jucr.opensolaris.org/pending/

In this repository, all the applications which have been submitted by the OpenSolaris community members, but are yet to be approved to be included in the contrib repository are available. More than 11,000 packages are available in this repository for download for free.

- http://opensolaris.homeunix.com:10906/

This repository has all multimedia packages for OpenSolaris. You can use this repository to download MP3 and video codecs as well as video players such as Mplayer can be downloaded from this repository for free. Check out http://tr.im/homeunix to learn more. Keep in mind that homeunix is a private site with very limited bandwidth. Prefer installing Mplayer using SVR4 packages.

Quick Tip

If you think that these snapshots will take too much space on your hard disk then you are wrong! Time Slider uses underlying ZFS filesystem to take and store snapshots which are usually very small in size. Check out interview section at http://thinkdigit.com/opensolaris to learn how ZFS snapshot works. If you want to learn more about Time Slider then check out http://tr.im/timeslider

3.4.4 Restricted IPS repositories

- https://pkg.sun.com/extra

This repository can be used to download some of the extra packages which are not freely redistributable. Important packages like Flash plug-in for Firefox can be downloaded through this repository. It's free to use, but like other Sun supported commercial IPS repositories, one needs to register with Sun Developer Network (which is also free) to access this repository. If you register your copy of OpenSolaris (registering OpenSolaris is free), you can use your Sun account to access this repository.

- https://pkg.sun.com/opensolaris/support/

This is a paid repository which is available only to those customer who buy support subscription for OpenSolaris. In order to check out what are the support options available for OpenSolaris, visit http://tr.im/osolsupport.

3.5 Ten things to install after installing OpenSolaris

By now, you would have installed OpenSolaris 2009.06 and would have notices some of its cool features. Now let's extend your OpenSolaris by installing some applications which will let you use it for general purposes, development purposes and most importantly, entertainment purposes and give a list of top ten applications to install along with the repository where you can find them.

Here are some applications which we will recommend you to install once you have installed OpenSolaris and connected it to the internet. If you use the web installer link, then you can directly install these applications using the Web Installer.

Mplayer and Songbird running on a OpenSolaris 2009.06 Desktop

#	Application	Repository	Description	Web Installer Link
1	OpenOffice	/extra	Productivity suite. It can open most of the Microsoft Office files such as DOC and PPT.	http://tr.im/openoffice
2	Java-Dev	/extra	It contains the java developer tools including JDK and Netbeans. Check out http://netbeans.org for more.	http://tr.im/javajdk
3	VirtualBox	/extra	Create and run virtual machine for Windows and Linux.	http://pkg.sun.com/extra you need to register first at http://pkg.sun.com
4	Sun Studio 12	/extra	Sun compilers for C and C++ with GUI based IDE.	http://tr.im/sunccpkg
5	GCC-Dev	/extra	GNU Compiler Collection tool chain. Needed for building many open source applications from source.	http://tr.im/gccpkg
6	AMP-Dev	/extra	Apache, MySQL and PHP development cluster to develop PHP based web applications.	http://tr.im/ampdevpkg
7	Transmission	/extra	A BitTorrent client.	http://tr.im/tbtpkg
8	WINE	/contrib	It lets you run some Windows files and software (EXE) on OpenSolaris.	http://tr.im/winepkg
9	Elisa Media Center	/extra	Its' a media center where you watch videos from your disk as well as from the internet. It can scan your hard disk for videos and music and create a library.	http://tr.im/elisapkg
10	GIMP image editor	/extra	Image editing tool.	http://tr.im/gimppkg

3.5.1 Installing SVR4 Packages

There are many software that you can download from the internet or distribute as PKG file that follows SVR4 style packaging. In this section, we will learn how to install PKG files by installing three packages that are needed to install some GNU-based applications on OpenSolaris.

Many GNU applications are distributed as SVR4 packages and require GNUbase (Basic GNU Libraries), GNU gettext and GNU libinconv to function. We will install these applications using their PKG files (which are bundled in the Digit DVD under gnu_essentials or you can download them from http://www.sunfreepacks.com) Follow these steps to complete the installation of these packages:

- Open a terminal window in the folder where you have the PKG file or use `cd` command to enter that directory.

- Execute `pkgadd -d <filename.pkg>` command to install the packages. For installing the three packages mentioned above, follow these instructions:

```
$ pfexec pkgadd -d GNUbase.pkg
$ pfexec pkgadd -d Asgettext-0.17.pkg
$ pfexec pkgadd -d ASlibiconv-1.12.pkg
```

Quick Tip
Check out www.thinkdigit.com/opensolaris for a tutorial on how to access these restricted repositories and configure IPS to download applications from them.

Make sure that you provide correct filename to the `pkgadd -d` command. If the PKG file is not present in the current working directory, then provide the complete path.

It is important that you add `/opt/gnu/bin` in your system path to run these applications directly. Follow these steps to add `/opt/gnu/bin` in your path.

Open a terminal and execute `$ gedit ~/.bashrc`. This will open a gedit window to edit your bashrc file.

Add this line at the end of your .bashrc file `export PATH=$PATH:/opt/gnu/bin`. Save the file and close the terminal.

Once you install these three applications, you can install many other GNU applications which are distributed as PKG files. Digit DVD has a large collection of PKG files that you can install.

3.6 Managing services on your system though SMF

Unix operating systems traditionally include a set of services or software programs that are not associated with any interactive user login. SMF (Service Management Facility) provides an infrastructure that augments the traditional UNIX startup scripts, init run levels, configuration files, as well as other services and software programs. Services listen for and respond to requests to perform certain tasks, for example:

- Delivering mail
- Responding to FTP requests
- Permitting remote command execution

Quick Tip
You can use Package Manager Web Installer to install packages just by clicking on appropriate links on a web page. Try http://tr.im/openoffice to install OpenOffice using the web installer. Check out http://pkg.opensolaris.org for searching and installing many more packages right out of Firefox!

SMF simplifies the management of these system services, by creating a supported, unified model for services and service management on each OpenSolaris system. The fundamental unit of administration in the SMF framework is the

service instance. Each service instance is named with a Fault Management Resource Identifier (FMRI). The FMRI includes the service name and the instance name. For example, the FMRI for the sendmail service is `svc:/network/smtp:sendmail`, where network/smtp identifies the service and sendmail identifies the service instance. Examples of other acceptable FMRIs include:

Quick Tip
Adobe Flash Player plugin for Firefox is available at https://pkg.sun.com/extra repository. Check out a screencast on how to install flash player in OpenSolaris at http://thinkdigit.com/opensolaris

- `svc://localhost/system/system-log:default`
- `svc:/system/system-log:default`
- `system/system-log:default`

Services can be enabled from the GNOME Desktop or by using the command line.

3.6.1 Enabling services from the desktop

You can enable and disable some services that are installed on an OpenSolaris system by using the Services application that is located in the GNOME desktop. Enabling and disabling services by using the Services application is equivalent to using the `svcadm` command. The services that can be viewed and managed from the desktop depend on the authorisations that are assigned to the user account and the service itself.

3.6.2 How to enable services from the desktop

Step 1: Choose `System > Administration > Services`.
The Services window is displayed. Services that are enabled have a check mark in the check box next to the service. If the service is disabled, the check box is blank.
Step 2: To enable a service, place a check mark next to the service, then save the changes. Services are disabled in the same way.

3.6.3 Enabling services from the command line

The `svcs` command is used to identify which services are installed on a system. The `svcadm` command is used to administer or change the state of a service. To configure SMF services, you must assume a role with appropriate privileges, such as the root or Primary Administrator role.

For example, if you have to check if Network Automagic (nwam) service is currently running or disabled, then open a terminal and execute:


```
$ svcs -a | grep nwam
  online          Jan_14    svc:/network/
physical:nwam
```

Quick Tip
Do not randomly enable or disable services from the desktop. Doing so can prevent an OpenSolaris system from booting.

If you want to list out all the services which are currently enabled, then run the following command:

```
  $ svcs -a | grep online | less
  online         Jan_14    svc:/system/svc/
restarter:default
  online          Jan_14    svc:/network/loopback:default
  online                Jan_14      svc:/network/datalink-
management:default
  online          Jan_14    svc:/system/identity:node
  online          Jan_14    svc:/system/metainit:default
  online               Jan_14      svc:/system/filesystem/
root:default
  online          Jan_14    svc:/system/scheduler:default
  online          Jan_14    svc:/system/cryptosvc:default
  online                  Jan_14       svc:/network/ipsec/
ipsecalgs:default
  online         Jan_14   svc:/system/boot-archive:default
  online               Jan_14      svc:/system/filesystem/
usr:default
  online         Jan_14   svc:/system/device/local:default
  online               Jan_14      svc:/system/filesystem/
minimal:default
  online          Jan_14    svc:/system/identity:domain
  [...]
```

Press q when you reach to the end of the list to exit. Same way when you want to find out all the services that are not running, replace online with offline in grep.

To enable apache (if you have installed amp-dev package) service

```
$ pfexec svcadm enable svc:/network/http:apache22
```

To disable, replace enable with disable.

To display the status of a current service execute svcs -l command. For example:

```
  $ svcs -l svc:/network/http:apache22
  fmri          svc:/network/http:apache22
  name          Apache 2.2 HTTP server
```

```
  enabled       true
  state         online
  next_state    none
  state_time    Thu Jan 14 16:04:18 2010
  logfile       /var/svc/log/network-http:apache22.log
  restarter     svc:/system/svc/restarter:default
  contract_id   70
  dependency        require_all/error   svc:/milestone/
network:default                             (online)
  dependency     require_all/none  svc:/system/filesystem/
local:default                          (online)
  dependency    optional_all/error svc:/system/filesystem/
autofs:default                 (online)
```

For more information check the main pages of `svcs` and `svcadm`.

3.7 Tips on troubleshooting an OpenSolaris startup

You can switch from the default graphical boot screen to the text boot screen. The ability to switch to the text boot screen can be useful if you suspect that the system startup process is not proceeding normally. The text screen might contain informational messages or a request for user input. Switching to the text boot screen has no impact on the boot sequence, other than how the information is displayed on the screen. Initialization of the operating system continues and completes as normal. To switch to a text boot, press any key a few seconds after the graphical boot screen appears, and the progress bar starts.

3.7.1 Boot modes

If the graphics card on your system is not supported, the system boots in console mode when you insert the Live CD and attempt to use the GUI installer. Should you encounter this problem, use the following procedure, which describes how to perform an installation by logging in to another system and running the GUI installer remotely.

If Your System Boots in Console Mode, you can install from the Live CD. To do this, you'll need two networked systems: the system on which the OpenSolaris Live CD was booted (target system) and a remote system from which the installation will be performed. Both systems must have network access. It is not required that the two systems be on the same subnet. However, the target system must be reachable from the remote system. Also, the remote system must be running an operating system that supports a

graphical desktop.

1. On the system to be installed, insert the Live CD, then boot the system.
2. At the console login, type the default login and password. The default user login and password for OpenSolaris is `jack`.
3. Assume the root role by using the `su` command and password. The password is `opensolaris`.

```
$ su root
```

Password: `opensolaris`

4. Enable the service for the ssh remote login program.

```
# svcadm enable ssh
```

5. Display the IP address that is assigned by DHCP to the target system.

```
# ifconfig -a
```

6. On the remote system, open a terminal window, then type:

```
$ ssh -X ip-address-of-target -l jack
```

where ip-address-of-target is the output of the `ifconfig -a` command that you ran on the target system. Running this command on the remote system opens a secure shell, enabling you to access the target system to use the GUI installer.

7. Assume the root role.

```
$ su root
```

Password: `opensolaris`

8. Run the GUI installer:

```
# /bin/gui-install
```

9. After the installation completes, reboot the target system.

If you are unable to log in to your installed system, use the following procedure.

Quick Tip

After switching from the graphical boot to a text boot, there is no way to switch back to the graphical boot screen. Wait for the graphical login screen to appear. If there is anything wrong, the text boot screen will show warning message and the graphical login screen will not appear.

{dev.works}

Don't forget to visit the definitive resource for programmers from Digit: dev.Works at devworks.thinkdigit.com

3.7.2 How to troubleshoot your OpenSolaris login

Troubleshooting login issues requires you to gain access to the system by booting in single-user mode, so you can troubleshoot the nature of problem and apply the appropriate solution. This procedure includes the steps for booting a system in single-user mode, as well as solutions to the most common login problems.

1. Boot the system in single-user mode.
a. When the boot sequence starts, and the GRUB menu is displayed, type `e` to display the GRUB edit menu.
b. Using the up or down arrow keys, select the kernel$ line, then type `e` to edit that entry.
c. Type `-s` at the end of the kernel$ entry.
d. Press `[Enter]` to go back to the previous screen.
e. To boot the system in single-user mode, type `b`.
2. When prompted, type an account name. The account name can be root, or any other privileged account, such as `jack` on the Live CD, or an account that you created during the installation.
3. Type the root password.

After the system has booted, depending on your particular situation, you can do any of the following:

Display the existing user accounts and roles:

```
-bash-3.2# cat /etc/user_attr
```

Delete a user account:

```
-bash-3.2# userdel username
```

Create a new user account:

```
-bash-3.2# useradd username
```

a. Assign a password for the user name.

```
-bash-3.2# passwd username
```

You will need to type the password twice.

b. Assign the root role to that user.

```
-bash-3.2# usermod R root username
```

4. To return to the installed system, type `exit`.

3.7.3 How to add the zfssnap role to the default user account

The default user account that is created when you install the OpenSolaris operating system includes the Primary Administrator profile. This account effectively has the same privileges as the root user. However, the default user account does not have the zfssnap role assigned to it, which is required to change Time Slider behaviour in the GNOME file browser. Before you can change Time Slider behaviour in the file browser, you must assign the zfssnap role to your default user account.

1. Choose `System > Administration > Users and Groups`.
2. Select your user account, then click the Properties button. The Settings

window for your user account is displayed.
3. To display the roles that are assigned to your user account, click the User Roles tab.
4. Place a check mark in the check box next to zfssnap, then click Ok to return to the Users and Groups window.
5. Save the settings.
6. To verify the role was assigned, open a terminal window and type:

```
$ roles
  root,zfssnap
```

3.8 Troubleshooting Device Drivers

If a device driver is missing, a peripheral device may not work properly. Therefore it is important to install drivers for the peripheral devices that you want to use.

3.8.1 Installing a driver package from the IPS repository

Run the Device Driver Utility. Choose `Applications > System Tools > Device Driver Utility`. If a highlighted entry indicates a driver is available in the IPS repository (driver available), the tool can download and install the driver package for you. Make sure your system has Internet Access. Right-click the highlighted entry. Select the Install Driver option from the popup that is displayed. Alternately, yo can install all missing drivers by clicking the Install All Drivers button.

3.8.2 Install a driver package not within the IPS repository

Run the Device Driver Utility. Choose `Applications > System Tools > Device Driver Utility`. If a highlighted entry indicates a third-party driver is available for that device (third-party), then the tool can display the URL to the recommended driver. Right-click the highlighted entry. Select the Show Details option from the popup that is displayed. At the top of the Details window, a URL to the recommended driver is displayed. To install the driver, follow the installation instructions that are included with the driver package.

In most cases, the driver is installed by copying the driver's binary file and its configuration file, if one exists, to the correct location on your system.

64-bit x86 drivers – Copy the driver to the `/kernel/drv/amd64/` directory. 32-bit x86 drivers – Copy the driver to the `/kernel/drv/` directory.

4 ZFS file system

A file system is responsible for managing files and operations that involve files. But even though most of the things we do on our computer involves a file system either directly or indirectly, most of us hardly pay attention to the file systems of our operating system. In this section, we will explore one of the most interesting aspects of OpenSolaris,which is its file system, the ZFS file system. Tools like Time Slider and Boot Environments make use of the capabilities ZFS file systems has, in order to provide the unmatched features in OpenSolaris. In the following sections, we will explore the ZFS file system, its features, usage, etc. by presenting relevant portions from the Solaris ZFS Administration Guide, published by Sun Microsystems Inc. This guide is also available on Digit DVD or you can download it from http://tr.im/zfsguide.

For trying out exercises in this chapter, we recommend you assume root user role. This can be done by executing `$ pfexec su` – in a terminal.

If you do not understand any term related to ZFS, then look for in the ZFS Terminology table later in the chapter.

4.1 What is ZFS?

The ZFS file system is a revolutionary new file system that fundamentally changes the way file systems are administered, with features and benefits not found in any other file system available today. ZFS has been designed to be robust, scalable, and simple to administer.

4.2 ZFS pooled storage

ZFS uses the concept of storage pools to manage physical storage. Historically, file systems were constructed on top of a single physical device. To address multiple devices and provide for data redundancy, the concept of a volume manager was introduced to provide the image of a single device so that file systems would not have to be modified to take advantage of multiple devices. This design added another layer of complexity and ultimately prevented certain file system advances, because the file system had no control over the physical placement of data on the virtualised volumes.

ZFS eliminates the volume management altogether. Instead of forcing you to create virtualised volumes, ZFS aggregates devices into a storage pool. The storage pool describes the physical characteristics of the storage (device

layout, data redundancy, and so on,) and acts as an arbitrary data store from which file systems can be created. File systems are no longer constrained to individual devices, allowing them to share space with all file systems in the pool. You no longer need to predetermine the size of a file system, as file systems grow automatically within the space allocated to the storage pool. When new storage is added, all file systems within the pool can immediately use the additional space without additional work. In many ways, the storage pool acts as a virtual memory system. When a memory DIMM is added to a system, the operating system doesn't force you to invoke some commands to configure the memory and assign it to individual processes. All processes on the system automatically use the additional memory.

4.3 Transactional semantics

ZFS is a transactional file system, which means that the file system state is always consistent on disk. Traditional file systems overwrite data in place, which means that if the machine loses power, for example, between the time a data block is allocated and when it is linked into a directory, the file system will be left in an inconsistent state. Historically, this problem was solved through the use of the `fsck` command. This command was responsible for going through and verifying file system state, making an attempt to repair any inconsistencies in the process. This problem caused great pain to administrators and was never guaranteed to fix all possible problems. More recently, file systems have introduced the concept of journaling.

The journaling process records action in a separate journal, which can then be replayed safely if a system crash occurs. This process introduces unnecessary overhead, because the data needs to be written twice, and often results in a new set of problems, such as when the journal can't be replayed properly. With a transactional file system, data is managed using copy on write semantics. Data is never overwritten, and any sequence of operations is either entirely committed or entirely ignored. This mechanism means that the file system can never be corrupted through accidental loss of power or a system crash. So, no need for a `fsck` equivalent exists. While the most recently written pieces of data might be lost, the file system itself will always be consistent. In addition, synchronous data (written using the O_DSYNC flag) is always guaranteed to be written before returning, so it is never lost.

4.4 Checksums and self-healing data

With ZFS, all data and metadata (which carries the information about the

nature of the data) is checksummed using a user-selectable algorithm. Traditional file systems that do provide checksumming have performed it on a per-block basis, out of necessity due to the volume management layer and traditional file system design. The traditional design means that certain failure modes, such as writing a complete block to an incorrect location, can result in properly checksummed data that is actually incorrect. ZFS checksums are stored in a way such that these failure modes are detected and can be recovered from gracefully. All checksumming and data recovery is done at the file system layer, and is transparent to applications. In addition, ZFS provides for self-healing data. ZFS supports storage pools with varying levels of data redundancy, including mirroring and a variation on RAID-5. When a bad data block is detected, ZFS fetches the correct data from another redundant copy, and repairs the bad data,replacing it with the good copy.

Quick Tip
Check out a video on how ZFS Self-Healing recovers data from a hard disk which has been smashed by a hammer at http://www.thinkdigit.com/opensolaris

4.5 Unparalleled scalability

ZFS has been designed from the ground up to be the most scalable file system, ever. The file system itself is 128-bit, allowing for 256 quadrillion zetabytes of storage. All metadata is allocated dynamically, so no need exists to pre-allocate inodes (inode is a structure which provide details about the file such as name and attributes) or otherwise limit the scalability of the file system when it is first created. All the algorithms have been written with scalability in mind. Directories can have up to 248 (256 trillion) entries, and no limit exists on the number of file systems or number of files that can be contained within a file system.

4.6 ZFS snapshots

A snapshot is a read-only copy of a file system or volume. Snapshots can be created quickly and easily. Initially, snapshots consume no additional space within the pool. As data within the active dataset changes, the snapshot consumes space by continuing to reference the old data. As a result, the snapshot prevents the data from being freed back to the pool.

ZFS snapshots include the following features:

- Persist across system reboots.
- The theoretical maximum number of snapshots is 264.
- Use no separate backing store. Snapshots consume disk space directly

from the same storage pool as the file system from which they were created.

- Recursive snapshots are created quickly as one atomic operation. The snapshots are created together (all at once) or not created at all. The benefit of atomic snapshot operations is that the snapshot data is always taken at one consistent time, even across descendent file systems.

ZFS Terminology	
alternate boot environment	A boot environment that is created by the lucreate command and possibly updated by the luupgrade command, but it is not currently the active or primary boot environment. The alternate boot environment (ABE) can be changed to the primary boot environment (PBE) by running the luactivate command.
checksum	A 256-bit hash of the data in a file system block. The checksum capability can range from the simple and fast fletcher4 (the default) to cryptographically strong hashes such as SHA256.
clone	A file system whose initial contents are identical to the contents of a snapshot.
dataset	A generic name for the following ZFS entities: clones, file systems, snapshots, or volumes. Each dataset is identified by a unique name in the ZFS namespace. Datasets are identified using the following format: pool/path[@snapshot] pool Identifies the name of the storage pool that contains the dataset path Is a slash-delimited path name for the dataset object snapshot Is an optional component that identifies a snapshot of a dataset
file system	A ZFS dataset of type filesystem that is mounted within the standard system namespace and behaves like other file systems.
mirror	A virtual device that stores identical copies of data on two or more disks. If any disk in a mirror fails, any other disk in that mirror can provide the same data.
pool	A logical group of devices describing the layout and physical characteristics of the available storage. Space for datasets is allocated from a pool.
primary boot environment	A boot environment that is used by the lucreate command to build the alternate boot environment. By default, the primary boot environment (PBE) is the current boot environment. This default can be overridden by using the lucreate -s option.
RAID-Z	A virtual device that stores data and parity on multiple disks, similar to RAID-5 (RAID stands for Redundant Array of Independent Disks)
resilvering	The process of transferring data from one device to another device is known as resilvering. For example, if a mirror component is replaced or taken offline, the data from the up-to-date mirror component is copied to the newly restored mirror component. This process is referred to as mirror resynchronization in traditional volume management products.
virtual device	A logical device in a pool, which can be a physical device, a file, or a collection of devices.
volume	A dataset used to emulate a physical device (like a hard disk). For example, you can create a ZFS volume as a swap device.

Snapshots of volumes cannot be accessed directly, but they can be cloned, backed up, rolled back to, and so on.

4.7 Simplified administration

Most importantly, ZFS provides a greatly simplified administration model. Through the use of hierarchical file system layout, property inheritance, and automanagement of mount points and NFS share semantics, ZFS makes it easy to create and manage file systems without needing multiple commands or editing configuration files. You can easily set quotas or reservations, turn compression on or off, or manage mount points for numerous file systems with a single command. Devices can be examined or repaired without having to understand a separate set of volume manager commands. You can take an unlimited number of instantaneous snapshots of file systems. You can backup and restore individual file systems. ZFS manages file systems through a hierarchy that allows for this simplified management of properties such as quotas, reservations, compression, and mount points. In this model, file systems become the central point of control. File systems themselves are very cheap (equivalent to a new directory), so you are encouraged to create a file system for each user, project, workspace, and so on. This design allows you to define fine-grained management points.

4.8 Getting Started with ZFS

In this section, we will show some simple commands which you can use in order to make use of ZFS file system.

So to make a 1 GB file, you would need to execute `$ mkfile 1g myfile`. To create a 100 MB file you will execute `$ mkfile 100m myfile`.

4.8.1 Creating a Basic ZFS file system

ZFS administration has been designed with simplicity in mind. Among the goals of the ZFS design is to reduce the number of commands needed to create a usable file system. When you create a new pool, a new ZFS file system is created and mounted automatically. The following example illustrates how to create a simple mirrored storage pool named tank and a ZFS file system named tank in one command. Assume that the whole of `/dev/dsk/c1t0d0` and `/dev/dsk/c2t0d0` are available for use. You can use two files (which are bigger than 128 MB) instead of the disks in order to try this out.

```
# zpool create tank mirror c1t0d0 c2t0d0
```

OpenSolaris and Solaris have a standard naming convention for storage

disk which is in the format cXtXdXpX. If you want to know the name of disk devices attached to your system, execute `$ format -e` in a terminal. This will list out all the disks along with any removable media.

The new ZFS file system, tank, can use as much of the disk space as needed, and is automatically mounted at /tank.

```
# mkfile 100m /tank/foo
# df -h /tank
Filesystem        size   used   avail  capacity
Mounted on
tank                     80G    100M          80G
1%     /tank.
```

Quick Tip
You do not need a hard disk or USB devices to try these things. Just make a file and you can use it with ZFS as a storage device. In order to create a file, use the following command:
$ mkfile <size of the file> <name of the file>

Within a pool, you will probably want to create additional file systems. File systems provide points of administration that allow you to manage different sets of data within the same pool. The following example illustrates how to create a file system named fs in the storage pool tank.

```
# zfs create tank/fs
```

The new ZFS file system, tank/fs, can use as much of the disk space as needed, and is automatically mounted at /tank/fs.

```
# mkfile 100m /tank/fs/foo
# df -h /tank/fs
Filesystem        size   used
avail  capacity     Mounted on
tank/fs                  80G
100M          80G          1%
/tank/fs
```

Quick Tip
Time slider in OpenSolaris makes use of ZFS Snapshots. Therefore in the beginning, there is hardly an overhead when Time Slider creates snapshots. But if you start running out of space, then you can delete some of the old snapshots to free up space.

4.8.2 Creating a ZFS Storage Pool

The previous example illustrates the simplicity of ZFS. The remainder of this chapter demonstrates a more complete example similar to what you would encounter in your environment. The first tasks are to identify your storage requirements and create a storage pool. The pool describes the physical characteristics of the storage and must be created before any file systems are created.

Follow the following steps to create a ZFS storage pool:

1. Become root or assume an equivalent role with the appropriate ZFS rights profile.
2. Pick a pool name
3. Create a pool

   For example, create a mirrored pool that is named tank.

   ```
   # zpool create tank mirror c1t0d0 c2t0d0
   ```

   If one or more devices contains another file system or is otherwise in use, the command cannot create the pool.
4. View the result

   You can determine if your pool was successfully created by using the zpool list command.

   ```
   # zpool list
   NAME        SIZE   USED   AVAIL  CAP    HEALTH ALTROOT
   tank                80G    137K          8 0 G
0%     ONLINE                -
   ```

4.8.3 Creating a ZFS file system hierarchy

After creating a storage pool to store your data, you can create your file system hierarchy. Hierarchies are simple yet powerful mechanisms for organizing information. They are also very familiar to anyone who has used a file system.

ZFS allows file systems to be organized into arbitrary hierarchies, where each file system has only a single parent. The root of the hierarchy is always the pool name. ZFS leverages this hierarchy by supporting property inheritance so that common properties can be set quickly and easily on entire trees of file systems.

Follow the steps to create ZFS file systems:

1. Become root or assume an equivalent role with the appropriate ZFS rights profile.
2. Create the desired hierarchy. In this example, a file system that acts as a container for individual file systems is created.

   ```
   # zfs create tank/home
   ```

   Next, individual file systems are grouped under the home file system in the pool tank.
3. Set the inherited properties.

   After the file system hierarchy is established, set up any properties that should be shared among all users:

   ```
   # zfs set   mountpoint=/export/zfs tank/home
   ```

```
  # zfs set    sharenfs=on tank/home
  # zfs set    compression=on tank/home
  # zfs get    compression tank/home
  NAME                PROPERTY                 VALUE
SOURCE
       tank/home          compression              on
local
```

A new feature is available that enables you to set file system properties when the file system is created. For example:

```
  # zfs create
  -o mountpoint=/export/zfs \
  -o sharenfs=on \
  -o compression=on tank/home
```

{dev.works}

Don't forget to visit the definitive resource for programmers from Digit: dev.Works at devworks.thinkdigit.com

4. Create individual file systems

   Note that the file systems could have been created and then the properties could have been changed at the home level. All properties can be changed dynamically while file systems are in use.

```
  # zfs create tank/home/bonwick
  # zfs create tank/home/billm
```

   These file systems inherit their property settings from their parent, so they are automatically mounted at /export/zfs/user and are NFS shared. You do not need to edit the `/etc/vfstab` or `/etc/dfs/dfstab` file.

5. Set the file system-specific properties.

   In this example, user `bonwick` is assigned a quota of 10 GB. This property places a limit on the amount of space he can consume, regardless of how much space is available in the pool.

```
  # zfs set quota=10G tank/home/bonwick
```

6. View the results

   View available file system information by using the zfs list command:

```
  # zfs list
  NAME USED   AVAIL  REFER  MOUNTPOINT
  tank                          92.0K         67.0G
9.5K   /tank
  tank/home                     24.0K         67.0G
8K     /export/zfs
  tank/home/billm               8K      67.0G
8K     /export/zfs/billm
  tank/home/bonwick     8K      10.0G         8K     /
```

```
export/zfs/bonwick
```

Note that the user `bonwick` only has 10 GB of space available, while the user `billm` can use the full pool (67 GB).

Now you have learned some basic commands to create and manage a ZFS zpool and file system. But it will be interesting to know what are the difference between the traditional file systems and ZFS file systems. In the following section, we would bring out some differences which ZFS has with other file systems.

4.8.4 ZFS file system granularity

Historically, file systems have been constrained to one device so that the file systems themselves have been constrained to the size of the device. Creating and re-creating traditional file systems because of size constraints are time-consuming and sometimes difficult. Traditional volume management products helped manage this process. Because ZFS file systems are not constrained to specific devices, they can be created easily and quickly, similar to the way directories are created. ZFS file systems grow automatically within the space allocated to the storage pool. Instead of creating one file system, such as `/export/home`, to manage many user subdirectories, you can create one file system per user. In addition, ZFS provides a file system hierarchy so that you can easily set up and manage many file systems by applying properties that can be inherited by file systems contained within the hierarchy.

4.8.5 ZFS space accounting

ZFS is based on a concept of pooled storage. Unlike typical file systems, which are mapped to physical storage, all ZFS file systems in a pool share the available storage in the pool. So, the available space reported by utilities such as `df` might change even when the file system is inactive, as other file systems in the pool consume or release space. Note that the maximum file system size can be limited by using quotas. Space can be guaranteed to a file system by using reservations. This model is very similar to the NFS model, where multiple directories are mounted from the same file system (consider `/home`).

All metadata in ZFS is allocated dynamically. Most other file systems pre-allocate much of their metadata. As a result, an immediate space cost at file system creation for this metadata is required. This behavior also means that the total number of files supported by the file systems is predetermined.

Because ZFS allocates its metadata as it needs it, no initial space cost is required, and the number of files is limited only by the available space. The output from the `df -g` command must be interpreted differently for ZFS than other file systems. The total files reported is only an estimate based on the amount of storage that is available in the pool. ZFS is a transactional file system. Most file system modifications are bundled into transaction groups and committed to disk asynchronously. Until these modifications are committed to disk, they are termed pending changes. The amount of space used, available, and referenced by a file or file system does not consider pending changes. Pending changes are generally accounted for within a few seconds. Even committing a change to disk by using fsync(3c) or O_SYNC does not necessarily guarantee that the space usage information is updated immediately.

4.8.6 Mounting ZFS File Systems

ZFS is designed to reduce complexity and ease administration. For example, with existing file systems you must edit the `/etc/vfstab` file every time you add a new file system. ZFS has eliminated this requirement by automatically mounting and unmounting file systems according to the properties of the dataset. You do not need to manage ZFS entries in the `/etc/vfstab file`.

4.9 ZFS Snapshots and Clones

By now, you already have an overview of ZFS snapshots. In this section, we will learn some commands to use snapshots and clones.

4.9.1 Creating and destroying ZFS snapshots

Snapshots are created by using the zfs snapshot command, which takes as its only argument the name of the snapshot to create. The snapshot name is specified as follows:

```
filesystem@snapname
volume@snapname
```

n the following example, a snapshot of tank/home/ahrens that is named friday is created.

```
# zfs snapshot tank/home/ahrens@friday
```

You can create snapshots for all descendent file systems by using the -r option. For example:

```
# zfs snapshot -r tank/home@now
# zfs list -t snapshot
```

```
NAME USED   AVAIL  REFER  MOUNTPOINT
tank/home@now                    0            -
29.5K         -
tank/home/ahrens@now     0            -      2.15M
-
tank/home/anne@now       0            -      1.89M
-
tank/home/bob@now                0            -
1.89M         -
tank/home/cindys@now             0            -
2.15M         -
```

Snapshots have no modifiable properties, nor can dataset properties be applied to a snapshot.

```
# zfs set compression=on tank/home/ahrens@tuesday
cannot set compression property for 'tank/home/
ahrens@tuesday': snapshot properties cannot be modified
```

Snapshots are destroyed by using the zfs destroy command. For example:

```
# zfs destroy tank/home/ahrens@friday
```

A dataset cannot be destroyed if snapshots of the dataset exist. For example:

```
# zfs destroy tank/home/ahrens
cannot destroy 'tank/home/ahrens': filesystem has
children
use '-r' to destroy the following datasets:
tank/home/ahrens@tuesday
tank/home/ahrens@wednesday
tank/home/ahrens@thursday
```

In addition, if clones have been created from a snapshot, then they must be destroyed before the snapshot can be destroyed.

4.9.2 Holding ZFS snapshots

If you have different automatic snapshot policies so that older snapshots are being inadvertently destroyed by ZFS receive because they no longer exist on the sending side, you might consider using the snapshots hold feature. Holding a snapshot prevents it from being destroyed. In addition, this feature allows a snapshot with clones to be deleted pending the removal of the last clone by using the `ZFS destroy -d` command. Each snapshot has an associated user-reference count, which is initialised to zero. This count

increases by one whenever a hold is put on the snapshot and decreases by one whenever a hold is released. In the previous Solaris release, snapshots could only be destroyed by using the `zfs destroy` command if it had no clones. In this Solaris release, the snapshot must also have a zero user-reference count. You can put a hold a snapshot or set of snapshots. For example, the following syntax puts a hold tag, keep, on `tank/home/cindys/snap@1`.

```
# zfs hold keep tank/home/cindys@snap1
```

You can use the `-r` option to recursively hold the snapshots of all descendent file systems. For example:

```
# zfs snapshot -r tank/home@now
# zfs hold -r keep tank/home@now
```

The above syntax adds a single reference, keep, to the given snapshot or snapshots. Each snapshot has its own tag namespace and tags must be unique within that space. If a hold exists on a snapshot, attempts to destroy that snapshot by using the `zfs destroy` command will fail. For example:

```
# zfs destroy tank/home/cindys@snap1
cannot destroy 'tank/home/cindys@snap1': dataset is busy
```

If you want to destroy a held snapshot, use the `-d` option. For example:

```
# zfs destroy -d tank/home/cindys@snap1
```

Use the `zfs holds` command to display a list of held snapshots. For example:

```
# zfs holds tank/home@now
NAME TAG    TIMESTAMP
tank/home@now     keep   Fri Oct 2 12:40:12 2009
# zfs holds -r tank/home@now
NAME TAG    TIMESTAMP
tank/home/cindys@now     keep   Fri   Oct   2   12:40:12 2009
tank/home/mark@now              keep   Fri    Oct    2 12:40:12 2009
tank/home@now                   keep   Fri    Oct    2 12:40:12 2009
```

You can use the `zfs release` command to release a hold on a snapshot or set of snapshots. For example:

```
# zfs release -r keep tank/home@now
```

If the snapshot is released, the snapshot can be destroy with the `zfs destroy` command. For example:

```
# zfs destroy -r tank/home@now
```

Two new properties identify snapshot hold information:

- The `defer_destroy` property is on if the snapshot has been marked for deferred destroy by using the `ZFS destroy -d` command. Otherwise, the property is off.
- The `userrefs` property is set to the number of holds on this snapshot.

4.9.3 Renaming ZFS snapshots

You can rename snapshots but they must be renamed within the same pool and dataset from which they were created. For example:

```
# zfs rename tank/home/cindys@083006 tank/home/cindys@today
```

In addition, the following shortcut syntax provides equivalent snapshot renaming syntax as the example above.

```
# zfs rename tank/home/cindys@083006 today
```

The following snapshot rename operation is not supported because the target pool and file system name are different from the pool and file system where the snapshot was created.

```
# zfs rename tank/home/cindys@today pool/home/cindys@saturday
cannot rename to 'pool/home/cindys@today': snapshots must be part of same dataset
```

You can recursively rename snapshots with the `zfs rename -r` command. For example:

```
# zfs list
NAME                     USED  AVAIL  REFER  MOUNTPOINT
users                    270K  16.5G    22K  /users
users/home               76K   16.5G    22K  /users/home
users/home@yest          0         -   22K  -
users/home/markm         18K   16.5G    18K  /users/home/markm
users/home/markm@yest 0       -    18K  -
users/home/marks         18K   16.5G    18K  /users/home/marks
users/home/marks@yest    0         -   18K  -
```

```
  users/home/neil          18K    16.5G        18K    /
users/home/neil
  users/home/neil@yest     0            -      18K    -
  # zfs rename -r users/home@yest @2daysago
  # zfs list -r users/home
  NAME                           USED   AVAIL  REFER
MOUNTPOINT
  users/home                     76K    16.5G
22K    /users/home
  users/home@2daysago      0            -      22K    -
  users/home/markm               18K    16.5G
18K    /users/home/markm
  users/home/markm@
  2daysago          0            -      18K    -
  users/home/marks               18K    16.5G
18K    /users/home/marks
  users/home/marks@
  2daysago          0            -      18K    -
  users/home/neil                18K    16.5G
18K    /users/home/neil
  users/home/neil@
  2daysago          0            -      18K    -
```

4.9.4 Displaying and accessing ZFS snapshots

You can enable or disable the display of snapshot listings in the `zfs list` output by using the `listsnapshots` pool property. This property is enabled by default. If you disable this property, you can use the `zfs list -t snapshot` command to display snapshot information. Or, enable the `listsnapshots` pool property. For example:

```
  # zpool get listsnapshots tank
  NAME        PROPERTY            VALUE        SOURCE
  tank        listsnapshots on                 default
  # zpool set listsnapshots=off tank
  # zpool get listsnapshots tank
  NAME        PROPERTY            VALUE        SOURCE
  tank        listsnapshots off
local
```

Snapshots of file systems are accessible in the `.zfs/snapshot` directory

within the root of the containing file system. For example, if `tank/home/ahrens` is mounted on `/home/ahrens`, then the `tank/home/ahrens@thursday` snapshot data is accessible in the `/home/ahrens/.zfs/snapshot/thursday` directory.

```
# ls /tank/home/ahrens/.zfs/snapshot
tuesday wednesday thursday
# zfs list -t snapshot
NAME                       USED  AVAIL  REFER  MOUNTPOINT
pool/home/anne@monday                0      -
780K   -
pool/home/bob@monday                 0      -
1.01M  -
tank/home/ahrens@tuesday          8.50K      -
780K   -
tank/home/ahrens@wednesday        8.50K      -
1.01M  -
tank/home/ahrens@thursday         0          -
1.77M  -
tank/home/cindys@today            8.50K      -
524K   -
```

You can list snapshots that were created for a particular file system as follows:

```
# zfs list -r -t snapshot -o name,creation tank/home
NAME                                          CREATION
tank/home/ahrens@tuesday                Mon  Aug  31
11:03 2009
tank/home/ahrens@wednesday      Mon Aug 31 11:03 2009
tank/home/ahrens@thursday               Mon  Aug  31
11:03 2009
tank/home/cindys@now                    Mon  Aug  31
11:04 2009
```

4.9.5 Snapshot space accounting

When a snapshot is created, its space is initially shared between the snapshot and the file system, and possibly with previous snapshots. As the file system changes, space that was previously shared becomes unique to the snapshot, and thus is counted in the snapshot's used property.

Additionally, deleting snapshots can increase the amount of space unique

to (and thus used by) other snapshots.

A snapshot's space referenced property is the same as the file system's was when the snapshot was created.

You can identify additional information about how the values of the used property are consumed. New read-only file system properties describe space usage for clones, file systems, and volumes. For example:

```
$ zfs list -o space
NAME AVAIL  USED   USEDSNAP      U  S  E  D  D  S
USEDREFRESERV       USEDCHILD
rpool              60.6G         6.37G         0
97K          0        6.37G
rpool/ROOT         60.6G         4.87G         0
21K          0        4.87G
rpool/ROOT/zfs1009BE     60.6G  4.87G    0        4.87G
0            0
rpool/dump         60.6G         1.50G         0
1.50G      0            0
rpool/swap         60.6G         16K    0
16K          0            0
```

4.9.6 Rolling back a ZFS snapshot

The `zfs rollback` command can be used to discard all changes made since a specific snapshot. The file system reverts to its state at the time the snapshot was taken. By default, the command cannot roll back to a snapshot other than the most recent snapshot. To roll back to an earlier snapshot, all intermediate snapshots must be destroyed. You can destroy earlier snapshots by specifying the `-r` option. If clones of any intermediate snapshots exist, the `-R` option must be specified to destroy the clones as well.

The file system that you want to roll back must be unmounted and remounted, if it is currently mounted. If the file system cannot be unmounted, the rollback fails. The `-f` option forces the file system to be unmounted, if necessary.

In the following example, the `tank/home/ahrens` file system is rolled back to the tuesday snapshot:

```
# zfs rollback tank/home/ahrens@tuesday
cannot rollback to 'tank/home/ahrens@tuesday': more
recent snapshots exist use '-r' to force deletion of the
following snapshots:
```

```
tank/home/ahrens@wednesday
tank/home/ahrens@thursday
# zfs rollback -r tank/home/ahrens@tuesday
```

In the above example, the wednesday and thursday snapshots are removed because you rolled back to the previous tuesday snapshot.

```
# zfs list -r -t snapshot -o name,creation tank/home/ahrens
NAME                                  CREATION
tank/home/ahrens@tuesday              Wed Aug 27 16:35 2008
```

4.9.7 Overview of ZFS clones

A clone is a writable volume or file system whose initial contents are the same as the dataset from which it was created. As with snapshots, creating a clone is nearly instantaneous, and initially consumes no additional disk space. In addition, you can snapshot a clone.

Clones can only be created from a snapshot. When a snapshot is cloned, an implicit dependency is created between the clone and snapshot. Even though the clone is created somewhere else in the dataset hierarchy, the original snapshot cannot be destroyed as long as the clone exists. The origin property exposes this dependency, and the `zfs destroy` command lists any such dependencies, if they exist. Clones do not inherit the properties of the dataset from which it was created. Use the `zfs get` and `zfs set` commands to view and change the properties of a cloned dataset. Because a clone initially shares all its disk space with the original snapshot, its used property is initially zero. As changes are made to the clone, it uses more space. The used property of the original snapshot does not consider the disk space consumed by the clone.

OpenSolaris Community
ZFS is open source software. If you want to be a part of ZFS Project and contribute then visit http://tr.im/zfscommunity

4.9.8 Creating and destroying ZFS clone

To create a clone, use the `zfs clone` command, specifying the snapshot from which to create the clone, and the name of the new file system or volume. The new file system or volume can be located anywhere in the ZFS hierarchy. The type of the new dataset (for example, file system or volume) is the same type as the snapshot from which the clone was created. You cannot create a clone of a file system in a pool that is different from where the

original file system snapshot resides.

In the following example, a new clone named `tank/home/ahrens/bug123` with the same initial contents as the snapshot `tank/ws/gate@yesterday` is created.

```
# zfs snapshot tank/ws/gate@yesterday
# zfs clone tank/ws/gate@yesterday tank/home/ahrens/bug123
```

In the following example, a cloned workspace is created from the `projects/newproject@today` snapshot for a temporary user as `projects/teamA/tempuser`. Then, properties are set on the cloned workspace.

```
# zfs snapshot projects/newproject@today
# zfs clone projects/newproject@today projects/teamA/tempuser
# zfs set sharenfs=on projects/teamA/tempuser
# zfs set quota=5G projects/teamA/tempuser
```

A clone can be destroyed by zfs destroy command.

```
# zfs destroy tank/home/ahrens/bug123
```

Clones must be destroyed before the parent snapshot can be destroyed.

4.9.9 Replacing a ZFS file system with a ZFS clone

You can use the `zfs promote` command to replace an active ZFS file system with a clone of that file system. This feature facilitates the ability to clone and replace file systems so that the origin file system becomes the clone of the specified file system. In addition, this feature makes it possible to destroy the file system from which the clone was originally created. Without clone promotion, you cannot destroy an origin file system of active clones.

Geek Hunt Contest

There are questions based on ZFS in the OpenSolaris Geek Hunt Contest at http://www.thinkdigit.com/opensolaris. Answer them and others to win an Acer Netbook!

In the following example, the `tank/test/productA` file system is cloned and then the clone file system, `tank/test/productAbeta`, becomes the `tank/test/productA` file system.

```
# zfs create tank/test
# zfs create tank/test/productA
# zfs snapshot tank/test/productA@today
# zfs clone tank/test/productA@today tank/test/
```

```
productAbeta
  # zfs list -r tank/test
  NAME        USED   AVAIL  REFER  MOUNTPOINT
  tank/test                314K   8.24G  25.5K    /tank/
test
  tank/test/productA       288K   8.24G  288K     /tank/
test/productA
  tank/test/productA@today       0            -
288K   -
  tank/test/productAbeta         0      8.24G  288K    /
tank/test/productAbeta
  # zfs promote tank/test/productAbeta
  # zfs list -r tank/test
  NAME              USED   AVAIL  REFER  MOUNTPOINT
  tank/test                316K   8.24G  27.5K  /tank/
test
  tank/test/productA             0      8.24G  288K   /
tank/test/productA
  tank/test/productAbeta   288K   8.24G 288K /tank/test/
productAbeta
  tank/test/productAbeta@today       0       - 288K -
```

In the above `zfs list` output, you can see that the space accounting of the original `productA` file system has been replaced with the `productAbeta` file system.

Complete the clone replacement process by renaming the file systems. For example:

```
  #   zfs   rename   tank/test/productA   tank/test/
productAlegacy
  # zfs rename tank/test/productAbeta tank/test/productA
  # zfs list -r tank/test
  NAME                           USED   AVAIL  REFER
MOUNTPOINT
  tank/test                316K   8.24G  27.5K  /tank/
test
  tank/test/productA       288K   8.24G  288K   /tank/
test/productA
  tank/test/productA@today       0       -      288K   -
  tank/test/productAlegacy0      8.24G  288K    /tank/
```

```
test/productAlegacy
```

Optionally, you can remove the legacy file system. For example:

```
# zfs destroy tank/test/productAlegacy
```

4.10 Conclusion

As you can see, there are so many things which you can do with ZFS file system on OpenSolaris. There are many more funky stuff which you can do with ZFS and we are leaving it for you to explore that for yourself. We recommend that you read the Solaris ZFS Administration Guide, published by Sun Microsystems Inc. which you can download from http://tr.im/zfsguide.

{dev.works}

Don't forget to visit the definitive resource for programmers from Digit: dev.Works at devworks.thinkdigit.com

5 DTrace

According to many, what makes a programmer is his/her ability to write optimised code that goes on to build better performing applications. This involves keeping many aspects of programming like memory consumption and I/O operations in mind while developing an application. While it is possible to achieve one level of optimization by skillfully writing code, but it becomes increasingly difficult as the size and the complexity of the application increases. Therefore developers often feel a need for a tool which could explain them how exactly their application is behaving while it is under execution.

There is a second aspect to consider as well. Imagine that you have a web application which has thousands of users connected to it, (say something like Gmail or Facebook). Now, if a bug is discovered in your application, you cannot play around with it while the application is running on the deployment machines (servers with which other users are connected) because of performance as well as security reasons. Most of the time, the bug is recreated in a development environment on to which the developers try to find and fix the problem, but it is a very time consuming and tedious process. Therefore, both system administrators as well as software developers feel a need for a tool, which can help them find out faults in their application, while the application is still running, without hampering the performance and compromising with security.

DTrace is an answer to all these questions. Its a tool which software developers can use to optimize and debug their programs and also use to understand the behavior of their application, while the system administrators can use it for finding problem points in an application while it is still running in production environments. In this chapter, we will like to bring out a detailed overview of DTrace. We will use the DTrace Quick Start Guide published by Sun Microsystems Inc. for this purpose. The complete guide can be downloaded from http://blogs.sun.com/observatory/entry/dtrace_quick_start_guide and it is also included in the book. For a deeper understanding of DTrace, we recommend that you read Solaris Dynamic Tracing Guide, which is in the Digit

Quick Tip

Before you begin with this chapter, we recommend that you get familiar with Unix Shell Scripting. You can learn basics of shell programming at http://tr.im/unixshell If you are already familiar with shell scripting them go ahead and enjoy the chapter!

DVD and can be download from http://tr.im/dtraceguide.

5.1 Overview

DTrace is a comprehensive dynamic tracing facility that is built into the Solaris OS and can be used by administrators and developers to examine the behavior of both user programs and of the operating system itself. With DTrace you can explore your system to understand how it works, track down performance problems across many layers of software, or locate the cause of aberrant behavior. DTrace is safe to use on production systems and does not require restarting either the system or applications.

DTrace dynamically modifies the operating system kernel and user processes to record data at locations of interest, called probes. A probe is a location or activity to which DTrace can bind a request to perform a set of actions, such as record a stack trace, a timestamp, or the argument to a function. Probes are like programmable sensors in your Solaris system. DTrace probes come from a set of kernel modules called providers, each of which performs a particular kind of instrumentation to create probes.

You can access DTrace capabilities by using the `dtrace` command or by using DTrace scripts. You must be the super user on the system to execute the `dtrace` command or DTrace scripts, or you must have appropriate DTrace privileges. DTrace includes a new scripting language called D that is designed specifically for dynamic tracing. With D, you can easily write scripts that dynamically turn on probes and collect and process the information. By sharing D scripts, you can easily share your knowledge and troubleshooting methods with others.

5.2 Introduction to D Scripts

DTrace works on an event/callback model: You register for an event and implement a callback that is executed when the event occurs. Examples of events include executing a particular function, accessing a particular file, executing an SQL statement, garbage collecting in JavaTM. Examples of data collected in the callback include function arguments, time taken to execute a function, SQL statements. In DTrace the event definition is called a probe, the occurrence of the event is called probe firing, and the callback is called an action. You can use a predicate to limit the probes that fire. Predicates are statements that evaluate to a boolean value. The action executes only when the predicate evaluates to true. A D script consists of a probe description, a predicate, and actions as shown below:

```
probe description
/predicate/
{
        actions
}
```

Quick Tip
You need to be root (or a user privileged with DTrace rights in RBAC) . You can be root by executing $ pfexec su -

When the D script is executed, the probes described in the probe description are enabled. The action statements are executed when the probe fires (the probe event occurs) and the predicate evaluates to true.

Consider the following simple D script:

```
syscall::write:entry
/execname == "bash"/
{
        printf("bash with pid %d called write \
                system call\n",pid);
}
```

In this D script, the probe description is `syscall::write:entry` which describes the write system call.

The predicate is `/execname == "bash"/`. This script checks whether the executable that is calling the write system call is the bash shell. The `printf` statement is the action that executes every time bash calls the write system call.

Predicates and action statements are optional.

- If the predicate is missing, then the action is always executed.
- If the action is missing, then the name of the probe that fired is printed.

In the following simple D script the predicate is missing, so the action is always executed. This script prints all the processes successfully started in the system.

```
proc:::exec-success
{
        trace(execname)
}
```

The following D script is more complex. This script shows all the files that are opened in the system.

```
syscall::open:entry
{
        printf("%s opened %s\n",execname,copyinstr(
```

```
arg0))
  }
```

5.2.1 Probe description

The probe consists of four fields separated by colon characters:

```
provider:module:function:name
```

provider	Required. Specifies the layer that is instrumented. For example, the syscall provider is used to monitor system calls while the io provider is used to monitor the disk I/O.
module	Optional. Describes the module that is instrumented.
function	Optional. Describes the function that is instrumented.
name	Optional. Typically represents the location in the function. For example, use entry to instrument when you enter the function.

Wild card characters such as * and ? can be used. Leaving a field blank is equivalent to using the * wild card. The following are a few examples.

Probe Description	Explanation
syscall::open:entry	Entry into the open() system call
syscall::open*:entry	Entry into any system call that starts with "open" such as open() and open64()
syscall:::entry	Entry into any system call
syscall:::	All probes published by the syscall provider
pid1234:libc:malloc:entry	Entry into the malloc() routine in the libc library in pid 1234
pid1234::*open*:entry	Entry into any function in pid 1234 that has "open" in its name
pid1234:::entry	Entry into any function in pid 1234, including the

main executable and any library

5.2.2 Predicate

The predicate is any D expression. You can construct predicates using the many built-in variables or arguments provided by the probe. The following table shows a few examples. The action is executed only when the predicate evaluates to true.

Predicate	Explanation
cpu == 0	True if the probe executes on cpu0
pid == 1029	True if the pid of the process that caused the probe to fire is 1029
uid == 123	True if the probe is fired by a process owned by userid 123
execname == "mysql"	True if the probe is fired by the mysql process
execname != "sched"	True if the process is not the scheduler (sched)
ppid !=0 && arg0 == 0	True if the parent process id is not 0 and the first argument is 0

5.2.3 Action

The action section is a series of action commands separated by semicolon characters (;). The following table shows a few examples. The action is executed only when the predicate evaluates to true.

Action	Explanation
printf()	Print something using C-style printf() command
ustack()	Print the user level stack
trace()	Print the given variable

5.2.4 Developing a D Script

The following dtrace command prints all the functions that process id 1234 calls. The -n option specifies a probe name to trace.

```
# dtrace -n pid1234:::entry
```

The following example shows how to use the above

dtrace command in a script. The -s option means compile the D program source file.

```
#!/usr/sbin/dtrace -s
/* The above line means that the script that follows
   needs to be interpreted using dtrace.
   D uses C-style comments.
*/
pid1234:::entry
{}
```

Replace the 1234 in the above command or script with the id of a process that you own that you want to examine. Remember to make your script file executable. Use Ctrl-C to return to the shell prompt.

Quick Tip
You can find the pid of a program by ruuning pgrep command. So if you want to know the pid of firefox, then open a terminal and run $ pgrep firefox and you will get the pid

You should see output similar to the following:

```
# dtrace -n pid23:::entry
dtrace: description 'pid23:::entry' matched 7954 probes
CPU     ID                    FUNCTION:NAME
  1 105673          set_parking_flag:entry
  1 105912                  queue_lock:entry
^C
#
```

On systems other than Solaris systems, you must be the super user on the system to execute the dtrace command or D scripts. On Solaris systems, you must be the super user on the system to use DTrace, or you must have appropriate DTrace privileges. You must be the root user to profile or trace processes owned by root. To grant DTrace privileges, become the super user and use the usermod command for the specified non-root user or edit the /etc/user_attr file directly. You should only need dtrace_user and dtrace_proc privileges.

To make the above script easier to reuse, modify the script to take the process id as a parameter, as shown below.

```
#!/usr/sbin/dtrace -s
pid$1:::entry
{}
```

You must provide one and only one argument when you invoke this script. Otherwise, dtrace will not execute. DTrace is dynamic: It shows events as they happen. If you probe a process that is not busy, you see a blank line until a matching event occurs. Try the command with a busy process such as a web browser process.

5.2.5 Using Aggregate Functions

If you specified a busy process, you probably received a very large volume of output. The D language aggregate construct helps you manage this output. Aggregations collect the output in tables in memory and output a summary. Aggregations have the following form:

```
@name[table index(es)] =aggregate_function()
```

The following is an example of an aggregate construct:

```
@count_table[probefunc] = count() ;
```

Add this construct to the action area of your script

```
#!/usr/sbin/dtrace -s
pid$1:::entry
{
        @count_table[probefunc] = count() ;
}
```

When you run the modified script, you see output similar to the following. Most of the data is omitted from this sample output.

```
# ./myDscript.d 23
dtrace: script './myDscript' matched 7954 probes
^C
  __clock_gettime                   5
  ioctl                             20
  mutex_lock                       767
  sigon                           1554
```

This script collects information into a table until you enter Ctrl-C. When you enter Ctrl-C, DTrace outputs the data that was collected during that time. The output table lists each function only one time, with the number of times that function was entered, from fewer to greater number of times entered. The variable probefunc is a built-in variable that holds the name

of the function in each event. The aggregation function count() increments the number of times the function was called. Other popular aggregation functions include average(), min(), max(), and sum(). The probes are created dynamically when the script runs. The probes are disabled automatically when the script stops. DTrace also automatically deallocates any memory it allocated.

The following script uses the probemod, probefunc table index to collect a table that shows both function name and library name:

```
#!/usr/sbin/dtrace -s
pid$1:::entry
{
          @count_table[probemod,probefunc]=count();
}
```

The library name is in the first column as shown in the following sample:

```
# ./myDscript.d 23
dtrace: script './myDscript' matched 7954 probes
^C
  libc.so.1              __clock_gettime            5
  libc.so.1               ioctl                    20
  libc.so.1              mutex_lock                767
  libc.so.1               sigon                   1554
```

5.2.6 Calculating Time Spent in Each Function

To determine how much time is spent in each function, use the DTrace built-in variable timestamp. Create probes in the entry and return of each function and then calculate the difference between the two timestamp values. The timestamp variable reports time in nanoseconds from epoch.

```
#!/usr/sbin/dtrace -s
pid$1:::entry
{
          ts[probefunc] = timestamp;
}
pid$1:::return
{
          @func_time[probefunc] = sum(timestamp \
                    - ts[probefunc]);
   ts[probefunc] = 0;
```

```
}
```

Note that the ts[] is an array, and D has automatically declared and initialized it for you. Setting the value of a variable to 0 instructs dtrace to recover the memory of the variable. The output gives the function name and the total time spent in that function during the time the script ran.

```
# ./myDscript.d 23
dtrace: script './myDscript' matched 15887 probes
^C
  enqueue                    2952
  mutex_lock                 3151262
  cond_timedwait             31282360711332
```

5.2.7 Ignoring Functions Already Entered

Notice that the cond_timedwait line in the previous example has a very large time number. This is because the cond_timedwait() function was already executing when the D script was started. To avoid this condition, add the following predicate to the return probe section. This predicate ignores a function return if there was no enter for that function.

```
pid$1:::return
/ts[probefunc] != 0/
{
          @func_time[probefunc] = sum(timestamp \
                    - ts[probefunc]);
          ts[probefunc] = 0;
}
```

As in C, you can omit the != 0 part and just use /ts[probefunc]/ as the predicate.

5.2.8 Handling Multithreaded Applications

Two threads could execute the same function at the same time and produce a race condition. To handle this case you need one copy of the ts[] construct for each thread. DTrace addresses this with the self variable. Anything that you add to the self variable is made thread local. The following script ignores functions that were already entered and handles multithreaded applications.

```
#!/usr/sbin/dtrace -s
pid$1:::entry
{
```

```
            self->ts[probefunc] = timestamp;
}
pid$1:::return
/self->ts[probefunc]/
{
            @func_time[probefunc] = sum(timestamp \
                        - self->ts[probefunc]);
            self->ts[probefunc] = 0;
}
```

5.2.9 Collecting Information About an Application

As an alternative to specifying a process id number to be traced, you can specify a command to be traced. When you specify a command to be traced, DTrace collects data on that command process and all of the child processes of that command. DTrace displays the collected data after the specified command exits. To collect data about a command, use the `-c` option to specify the command on the `dtrace` command line or when you invoke your D script. If you use a D script, use the `DTrace $target` macro inside the script to capture the process id of the argument to the -c option on the command line.

The following script counts the number of times libc functions are called from a specified command:

```
#!/usr/sbin/dtrace -s
pid$target:libc::entry
{
            @[probefunc]=count();
}
Use the -c option to specify the target command:
# ./myDscript.d -c "man dtrace"
```

In this example, the first output you see is the dtrace man page. When you press the q key, you see output such as the following. Most of the output is omitted in this example.

```
dtrace: script ’./myDscript.d’ matched 2914 probes
dtrace: pid 1111 has exited
  ___lwp_private              1
  strncpy                   125
  strlen                   7740
```

5.3 DTrace for Developers

The providers discussed in this section collect types of information that often are most interesting to application developers: syscall, proc, pid, sdt, vminfo. Use these providers to observe running processes as well as process creation and termination, LWP creation and termination, and signal handling. This section focuses on the pid provider.

5.3.1 The pid Provider

The pid provider instruments the entry and return from any user level function in a running process. With the pid provider you can trace any instruction in any process on the system. The name of the pid provider includes the process id of the running process that you want to examine.

5.3.2 The syscall Provider

The syscall provider reports information about system calls made. The following script displays the stack trace when a program makes a write() system call.

```
#!/usr/sbin/dtrace -s
syscall::write:entry
{
  @[ustack()]=count();
}
```

5.3.3 The sysinfo Provider

The sysinfo provider reports information about kernel statistics that are classified by the name sys. These kernel statistics provide the input for system monitoring utilities such as mpstat. The following script counts the number of times various processes get to run in the CPU. The sysinfo:::pswitch event occurs when a process is switched to run on the CPU.

Enter Ctrl-C to display the results from running this script.

```
#!/usr/sbin/dtrace -s
sysinfo:::pswitch
{
          @[execname] = count();
}
```

5.3.4 The proc Provider

The proc provider reports information about processes. The following script

displays the process name, process id, and user id when a new process is started in the system. The proc:::exec-success event occurs when a new process is started successfully.

The -q option suppresses output such as column headings and number of probes matched.

```
#!/usr/sbin/dtrace -qs
proc:::exec-success
{
          printf("%s(pid=%d) started by uid \
                    - %d\n",execname, pid, uid);
}
```

5.4 DTrace for Web 2.0 Applications

Digit Subscription

Subscribe to the Digit developer newsletter at www.thinkdigit.com/newsletter

The deployment stack for a typical Web 2.0 application is becoming increasingly complicated. The stack has JavaScript on the browser, multiple layers of API (for example, OpenSocial), an application server, a web server, a database, native code such as C/C++ or Java, and the operating system. A click on a web page can exercise multiple layers. Experts in one area might not be experts in another area, making performance tuning even more difficult. Each layer listed above has good debugging tools. Database administrators have GUI and command line tools to observe every detail of the database. Language debuggers such as gdb do a good job of providing visibility into both scripting and native languages in the stack. Tools such as JProbe, JMeter, and VisualVM provide visibility into Java code. Tools such as vmstat, iostat, mpstat, and truss provide visibility into the operating system.

These tools are important in performance tuning and understanding each layer in isolation. However, these tools do not provide insight into the entire system and interaction between the different layers. For example, these tools cannot help find the database queries that are executed when a user clicks a

JavaScript button on a browser. Also, some of these tools are not usable in production. For example, truss can slow down your application too much to be usable in production. Developers typically resort to creating custom debug code to address these issues. These pieces of code can be seen in debug enabled versions of applications. These debug versions of applications need special flags and reduce performance of the applications. Therefore, debug

enabled applications are not normally used in production.

DTrace addresses this problem with dynamic instrumentation. How do you wish you could debug and performance tune your web 2.0 applications? What if you could dynamically insert code into a live running application and collect data at the point of instrumentation? What if you could turn these instrumentations on and off dynamically? What if you could use the same instrumentation at every layer of your application? What if this instrumentation supports popular languages such as C, C++, Java, PHP, Python, Ruby, JavaScript? These capabilities are exactly what DTrace provides for you.

DTrace enables you to observe every layer of your application infrastructure. It enables you to collect data for a few seconds (incurring a smaller performance penalty) and turn off data collection dynamically – without restarting applications. You do not need to put new code into your application to use DTrace. DTrace can observe fully performance tuned code – no `-g` option is needed.

5.5 Observing multiple layers

D scripts can span multiple layers. The following script reports how much time you are spending on the different layers in an AMP stack (Apache/MySQL/PHP). The output is `%s` of time spent in Apache, Java, MySQL, the browser, and the operating system. This script is like a TOP tool for AMP.

```
#!/usr/sbin/dtrace -qs
BEGIN
{
     total=mysqlcnt=httpcnt=phpcnt=javacnt=ffxcnt=oth
ercnt=0;
      printf("%10s %10s %10s %10s %10s %10s\n","%
MYSQL","% APACHE",\
        "% FIREFOX","% PHP","% Java","% OTHER");
}
php*:::request-startup
{
     inphp[pid,tid]=1;
}
php*:::request-shutdown
{
     inphp[pid,tid]=0;
```

{dev.works}

Don't forget to visit the definitive resource for programmers from Digit: dev.Works at devworks.thinkdigit.com

```
  }
  profile-1001
  {
       total++;
       (execname=="mysqld")?mysqlcnt++:\
            (execname=="httpd")?(inphp[pid,tid]==1?phpcn
t++:httpcnt++):\
             (execname=="java")?javacnt++:\
           (execname=="firefox-bin")?ffxcnt++:othercnt++;
  }
  tick-30s
  {
         printf("%10s  %10s  %10s  %10s  %10s  %10s\n","%
MYSQL","% APACHE",\
    "% FIREFOX","% PHP","% Java","% OTHER");
  }
  tick-2s
  {
    printf("%10d %10d %10d %10d %10d %10d\n",mysqlcnt*100/
total,\
         httpcnt*100/total,ffxcnt*100/total, phpcnt*100/
total,\
             javacnt*100/total,othercnt*100/total);
       total=mysqlcnt=httpcnt=phpcnt=f                fxcn
t=javacnt=othercnt=0;
  }
```

Quick Tip

You would need to install the amp-dev package from the Package Manager in OpenSolaris 2009.06 in order to execute these applications. Also, you can download these codes from the DTrace Quick Start Guide Wiki at http://wikis.sun.com/display/DTrace/DTrace+Quick+Start+Guide

5.6 Observing One Application

In the previous example you saw how to use DTrace to observe many layers of the stack at production. The examples in this section show how to use DTrace to observe one particular application. With a little knowledge of the application, you can develop D scripts that provide important information about the application while the application is running.

5.7 Observing the MySQL database

Quick Tip
For more such Do-It-Yourself exercises on Web 2.0 applications, read DTrace Quick Start Guide which is bundled with the DVD and also available for download from http://blogs.sun.com/observatory/entry/dtrace_quick_start_guide

Consider the database MySQL, for example. MySQL is open source, and you can easily discover that the name of the function that is called when a particular SQL statement is executed is dispatch_command(). You can also easily determine that the SQL statement is passed as a string in the third argument. With only this knowledge you can write the following very simple D script to print out the SQL that are executed in a live running instance of MySQL.

```
#!/usr/sbin/dtrace -qs
#pragma D option strsize=1024
pid$1::*dispatch_command*:entry
{
                printf("%d::%s\
n",tid,copyinstr(arg2));
}
```

DTrace Community
DTrace is an open source software and if you want to be a part of this project then visit http://tr.im/dtracecommunity Anyone can contribute!

The option strsize is used to increase the size of strings in D to handle longer SQL statements.

5.8 Conclusion

As you can see, DTrace can help you take application development to a whole new level! We recommend that you try simple scripts yourself and discuss dtrace at http://thinkdigit.com/forum.

6 Contributing with SourceJuicer

Contributing to an open source project can be a very rewarding experience. If you decide to contribute code to a open source project, you can do it in many ways. For example, you can contribute by developing new application, add features to an existing application or port an application to a platform so that people can install and use it. In this chapter, we will learn how to port applications on OpenSolaris, so that you can port your own applications on it, or port existing open source applications so that many other people can use it by downloading and installing it through the OpenSolaris IPS repositories. There are thousands of applications which are out there for you to port, and it will be an easy and constructive first step if you want to become an open source contributor. In this chapter we will follow the SourceJuicer How-To guide to learn how to port your own packages in OpenSolaris.

You can download the PDF version of this How-To guide for free from http://tr.im/sjhowto. SourceJuicer is an automatic build and package service for OpenSolaris. In particular, it allows OpenSolaris developers to submit a build recipe to the SourceJuicer build service which will be automatically compiled and an IPS package created and published to a pending repository. Once the IPS package has been published, peer developers called 'Approvers' can review the package's contents, test it out and approve it to be published into an IPS repository for other users to install. The build recipe that a developer submits takes the form of a 'spec file'. Some developers may be familiar with spec files from the RPM package management system. The spec files used for OpenSolaris are similar, but have a few differences to help provide package meta-data during the creation process. To visit SourceJuicer, browse to http://jucr.opensolaris.org/.

6.1 IPS repositories

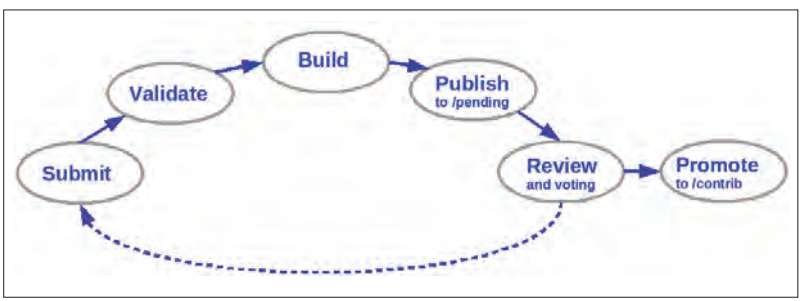
IPS repositories store a collection of software packages that can be installed through the Package Manager utility on OpenSolaris. Each software package has a name, a description, a license, contents, and a list of prerequisite software packages which are necessary for this package to install and run correctly. This How To guide will explore two different IPS repositories, `/pending` and `/contrib`, and their involvement in the SourceJuicer software package life-cycle.

Name	Description
http://jucr.opensolaris.org/pending/	The /pending repository is where all successful SourceJuicer submissions are initially published, prior to review by Approvers. It is a temporary repository for new software packages, prior to their promotion to /contrib.
http://pkg.opensolaris.org/contrib/	Once a software package has been reviewed by Approvers, and tested, it will be promoted across into the /contrib repository for wider use by OpenSolaris users.

6.2 Review cycle

Each submission to SourceJuicer goes through an important review cycle, as a new software package gets created, reviewed and migrated to the `/contrib` repository.

Once a spec file recipe has been submitted it's automatically checked for syntax errors, before being put in a queue for an Approver to validate the submission. Validation ensures that the package contains the correct licence and meta-data information. Upon successful validation, SourceJuicer's automatic build service then tries to create an IPS package. If successful, the package will be automatically published to the `pending/` repository. If any errors occur, the developer will be notified that the build has failed. The developer can then make the appropriate fixes and re-submit their spec file. The next phase in the life-cycle is the review phase. A package is reviewed to ensure the package meta-data information as well as the package



SourceJuicer Review Cycle (Courtesy : http://jucr.opensolaris.org/)

contents are a high enough quality to be ready to promote. Any member of the SourceJuicer community may participate in reviews, but in order for the package to be promoted to `/contrib`, it must have a positive (+1) vote from two Approvers and no negative (-1) votes.

6.2.1 Example: RoundCube

RoundCube is a webmail application, written in PHP, that provides an e-mail client in a web browser. It uses a database for back-end storage, and in this example we will be using the MySQL database.

6.2.2 Spec File

The spec file for RoundCube can be divided into a number of sections:

1. Information: package name, description, sources and other package meta-data
2. Preparation: when a source tarball gets unpacked
3. Build: how the software is compiled
4. Install: how the software is installed onto the system
5. Clean up: used to clean up the developer build environment
6. Package contents: define what files are to be included in the package
7. Log of changes: list the changes that have been made to the spec file over time

For this example we will take a very simplistic approach with RoundCube, and install the software into the Apache document root, /var/apache2/2.2/htdocs/ for later configuration.

```
#
# Spec file for package roundcubemail
#
# Copyright 2009 Sun Microsystems, Inc.
# This file and all modifications and additions to the pristine
# package are under the same license as the package itself

#
# Owner: gman
%include Solaris.inc
# Information
Name:           roundcubemail
```

```
  Summary:        Round Cube Web Mail
  Version:        0.2.1
  License:        GPLv2
  Source:         http://easynews.dl.sourceforge.net/
sourceforge/roundcubemail/%{name}-
  %{version}.tar.gz
  URL:            http://roundcubemail.sourceforge.net/
  Group:          Networking/Mail
  Distribution:   OpenSolaris
  Vendor:         OpenSolaris Community
  BuildRoot:      %{_tmppath}/%{name}-%{version}-build
  SUNW_Copyright: %{name}.copyright

  # Package dependencies
  %include default-depend.inc
                SUNWapch22
  Requires:
  Requires:       SUNWapch22m-php52
  Requires:       SUNWmysql

  # Additional package meta-data
  Meta(info.upstream):       http://roundcubemail.
sourceforge.net/
                           Glynn Foster
  Meta(info.maintainer):
  Meta(info.repository_url): http://roundcubemail.
sourceforge.net/
  Meta(info.classification): org.opensolaris.
category.2008:Web Services/Communications

  %description
```

RoundCube Webmail is a browser-based multilingual IMAP client with an application-like user interface. It provides full functionality you expect from an e-mail client, including MIME support, address book, folder management, message searching and spell checking. RoundCube Webmail is written in PHP and requires the MySQL, PostgreSQL or SQLite database. The user interface is fully skinnable using XHTML and CSS 2.

```
  # Preparation
```

```
  %prep
  %setup -q

  # Build
  %build

  # Installation – for simplicity, install into Apache
document root

  %install
  rm -rf $RPM_BUILD_ROOT
  mkdir   -p   $RPM_BUILD_ROOT/var/apache2/2.2/htdocs/
roundcubemail
  cp   -pr   *   $RPM_BUILD_ROOT/var/apache2/2.2/htdocs/
roundcubemail

  # Clean up
  %clean
  rm -rf $RPM_BUILD_ROOT

  # Package contents
  %files
  %defattr (-, root, bin)
  %dir %attr (0755, root, sys) %{_var}
  %{_var}/apache2/2.2/htdocs/roundcubemail/*

  # List of changes
  %changelog
  * Wed May 13 2009 - Glynn Foster <glynn.foster@sun.
com>
  - Add setup phase to unpack tarball
  * Tue May 12 2009 - Glynn Foster <glynn.foster@sun.
com>
  - Initial version
```

As you will notice from the above spec file, it is divided into the different sections listed above. In the information section, we specify the package name, description, version, location of the source, license, any software packages that we are dependent on, and additional package meta-data that

allows us to correctly classify the package according to its functionality.

The preparation section details how the sources are unpacked, applying any additional source patches where necessary. For this section we are using default commands which are known to handle sources that are packed using the tar, gzip and bzip2 utilities.

The build section details how the sources are compiled into binaries. In this case, because this is a PHP web based application, no compilation is necessary.

In the install section we install the resulting binaries into a temporary location to allow us to add them to a package while assuring that stray files, directories and permissions aren't included. As you will notice, we first make sure that temporary location is empty by making sure to remove any older files if they exist. In this case, we are simply copying over the sources files to the Apache document root in a default OpenSolaris installation.

The package contents section is the final important section of the spec file. This is where we specify which files should be included in the package, where they should be installed and what file permissions they should have. This How-To will avoid low level detail about the spec file format. If you would like to learn more about spec files, including the pkgbuild utility, see the Software Porters community on OpenSolaris at http://tr.im/swporters/.

6.3 Introducing SourceJuicer

To use the SourceJuicer automatic build and package service, an OpenSolaris account is required. You can register for an account by navigating to the "Log in" tab in the upper right corner of the SourceJuicer home page.

6.3.1 Submitting a Spec File

Once you have successfully logged in, you may begin submitting a new package. Navigate to the "Submit" tab. Each submission must have a unique identifier, a spec file and a copyright file. Optional patches and additional sources can also be specified in the submission. In this example, we have used the RoundCube spec file listed above, along with another file that details out the terms of the license for this piece of software. The spec file should have the same file name as the unique identifier. For example, roundcubemail.spec.

SourceJuicer Homepage

6.3.2 Validation of syntax and fixing it

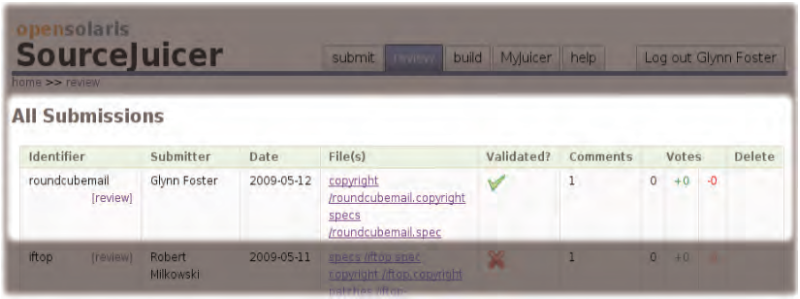
After you submit your request, a quick syntax validation check will be performed on your spec file and you will be notified of any failures, with a re-direct to the details page where you can fix the error. In the following case, the validation check showed there was a missing "URL" identifier. You can easily fix this by using the "Re-submit" tab further down the page to resubmit the spec file with an accompanying comment. If you need assistance from other SourceJuicer developers, you may simply add a comment using the "Comment" tab. Once all the errors have been corrected, you can now proceed to the next step.

6.3.3 Validation by approver

The next important step is for a package to be validated by an Approver. The Approver will do a quick check to make sure the package is what it says it is. He/she will make sure that the correct licence and copyright information have been specified. Once the Approver is satisfied, your package will be queued in the automatic build service.

6.3.4 Building the package

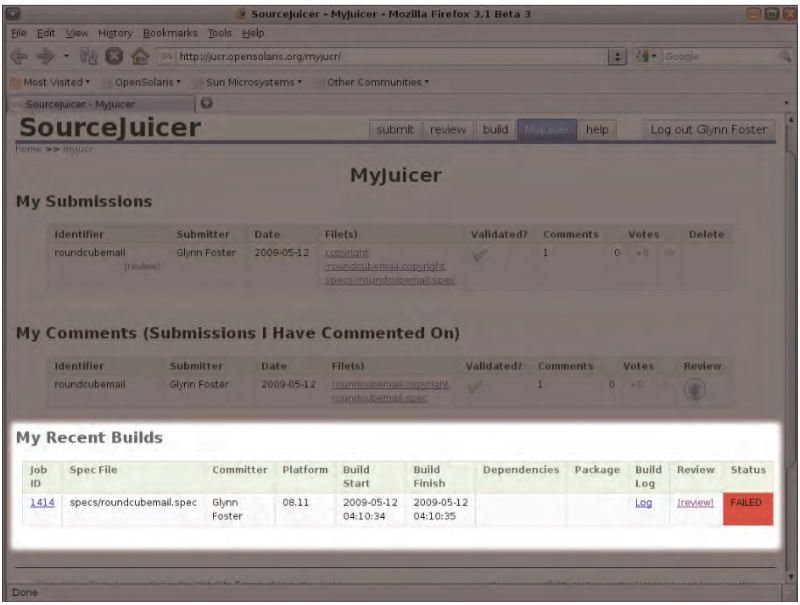
Now that the package has been validated, it can move on to be built by the automatic build service. If you have correctly specified software package dependencies in your spec file, SourceJuicer will automatically install the packages that your package requires. Provided you have correctly written



The RoundCube package has been validated

your spec file, the resulting output will be an IPS package published to the `/pending` repository. However, sometimes there will be issues with the creation of your package. The compile may fail, errors can occur while applying source patches or determining package contents.

The "build" tab lists the various packages that have been built recently, including those that are in the process of being built. You can check on how



MyJuicer showing the build of roundcubemail has failed and needs to be fixed

your build is progressing by viewing the build log. If the build succeeds, your package will be published and the status will be updated to "PASSED". However, if the build fails for any reason, it will be flagged as "FAILED" and you will need to correct the build failures.

6.3.5 Waiting for the Vote

Once you have successfully built and published your package to /pending, the package can be tested by a wider community and a more rigorous review can take place. Not only does the package need a correct name, description and license, but community reviewers should also check the various files your package installs and the general stability of the software.

More Information	
SourceJuicer Home Page	http://jucr.opensolaris.org/
OpenSolaris Software Porters Community	http://www.opensolaris.org/os/community/sw-porters
SourceJuicer Help	http://jucr.opensolaris.org/help/main/
OpenSolaris Software Porters Forum	http://tr.im/sjforum
OpenSolaris Homepage	http://www.opensolaris.com

A package can be installed by adding the /pending repository to the list of repositories on your OpenSolaris system. You can do this by launching Package Manager and using the "Manage Repositories" menu entry . Once you have added the /pending repository, you can install your new package simply by searching for the package name and filtering your result. As the package is reviewed, you may be asked to make modifications to your package either through a series of changes to the spec file or by applying additional source patches. Once the package is to their level of satisfaction, you will need a positive (+1) vote from two Approvers with no negative (-1) votes. You can interact with the Approvers using the "Comments" form in the review section of your package.

6.3.6 Publishing to /contrib

At regular intevals, packages with qualifying votes will be promoted to the /contrib repository. The administrators of SourceJucr will organize this automatically for you. Congratulations, you have successfully contributed a new package to OpenSolaris!

{dev.works}

Don't forget to visit the definitive resource for programmers from Digit: dev.Works at devworks.thinkdigit.com

7 Careers in OpenSolaris

Accumulating knowledge to quench your thirst of curiosity can be a regarding experience, but eventually everyone gets hungry. You may get great open source software for free, but a good meal still costs real money and at the end of the day we all have bills to pay. In this chapter, we would like to suggest some ways through which you can contribute to open source and build a career out of open source development. This chapter is specially relevant to high school and college students who are interested in making a career in IT world and who can use open source projects as learn many things by doing it rather than just reading about it!

7.1 Career as a Developer

At first, lets list out what are some of the skills employers would look for in a candidate for a software engineering role which involves programming or software design:

- Good knowledge of programming
- Good understanding of software engineering practices
- Ability to work in teams
- And with teams now going multinational, ability to work with people of different nationality and culture

Now lets examine how contributing to open source builds all these skills in you. The process of open source development involves lots of things. If you decide to contribute code to any popular existing open source project, or start a project of your own (which a sizable number of people should use of course!), you will have to go through a strong peer review of your code. You do not have to be a native speaker of C or C++ to be a open source developer, but you will need to have a flair for writing good quality code for it to get it added to the project. This means that you need to have good understanding of not only the syntax of a language, but also a grip on good programming practices and coding style. So when your resume says that you have contributed to a major open source project, the employers can be confident that you have good programming skills as well as good understanding of software engineering practices.

Usually, many contributors participate in a open source project and contributing to involves very strong teamwork. Many people work together to bring out a complete open source product. So in this process, you get

the experience of working in teams. Also, open source projects attract contributors from all across the globe and so working on these projects is one of the very effective channel of learning how to work in a multicultural work environment. The employers value this skill in individuals and your resume always stand out when they see your credentials in a open source project.

Other benefits of working in open source involves making contacts. When you participate in discussion forums, your questions and responses are read by many other community members and you gain respect in the community for your participation. People in the community start recognizing you and you also get to meet new people, who are aware of your skills. Many of the community members may be working for different organisation, and would recommend you for a relevant job opening in their organization. Once you get to know other members in the community very well, they may even refer you to their friends in the industry. Getting a job, and that too a well paying one, would be least of your problems if you have substantial contributions to open source.

Quick Tip

QuickTip: Whenever posting to a discussion forum, please maintain the forum etiquette. Using SMS like language is frowned upon, and asking questions, which has already been asked before shows that you have not done your home work. Make sure that you google for the answer you are looking for and then only throw it open for discussion in a forum, when you do not find it elsewhere.

OpenSolaris community is a good choice to start contributing to open source. The community is very well structured, and http://www.opensolaris.org is a common home for the community. Almost all of projects of OpenSolaris community is hosted here and so it makes it very simple for people to find where to start and how to begin. The community members are also very friendly and there are many support groups available. Biggest challenge one faces in contributing to open source is in understanding the community process. In OpenSolaris community, every process is well documented and every project and contribution is adjudged only on its merits in a democratic way. Create an account at http://www.opensolaris.org and experience it for yourself!

7.2 Career as a System / Network Administrator

If you are not one of those is deeply interested in writing code, but you like managing networks, security and systems, then a career in system administration or network administration will be a good choice. This will require you to have a good command over networking or system administration which you can learn by reading documentations and

Quick Tip
In a global survey, average annual salary of Sun Certified Solaris Administrators with relevant experience is reported to be approximately Rs. 37,50,000 (82,000 US Dollars). To read more about this survey, goto http://tr.im/whycert

experimenting different things with your own system.

Solaris is one of the most popular platform for servers and there are millions of servers running on Solaris. Many major banks, telecommunication companies, as well as many other web 2.0 companies run on Solaris. So there is a very strong demand for skilled Solaris administrators who have good knowledge of tools like DTrace and ZFS. So you get a hang of using OpenSolaris well, you can have a very flourishing career as a system administrator or network administrator.

But unlike contributing code, where there is a definite way to exhibit one's technical skills, there are not many ways where you can showcase your skills in use of Solaris. But there is a way out! You can take up the Solaris Certification Examination which is a standardized test (like GMAT, GRE) leading to certification as Solaris Administrator and Network Administrator. You can learn more about Solaris Certification at http://tr.im/solariscert.

7.3 Web Resources for Students

There are many social networking sites like LinkedIn (http://linkedin.com) where you can make your professional accomplishments count. But in this section we will cover two web services which are specifically for open source developers.

7.3.1 Open Source University Meetup (OSUM)

The first website we will feature in this section is Open Source University Meetup (OSUM, pronounced "awesome") (http://osum.sun.com) which is designed specially for students and student developers. This is a social networking website, where any educational institution

Open Source University Meetup Homepage

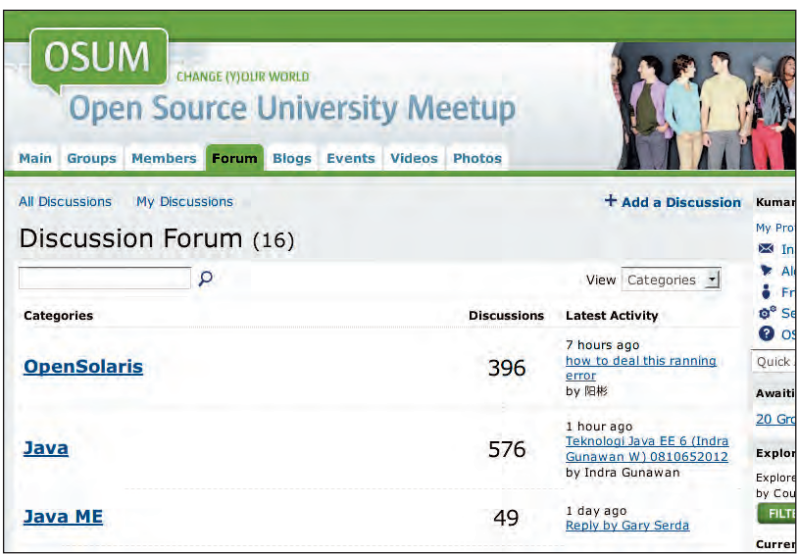
can create a group (Meetup), and invite fellow friends.

There you will find many resources like webinars and collection of tutorials. There are thousands of groups in OSUM, including those of many Indian universities. You can join these groups to meet fellow students who are interested in open source.

OSUM webinars are presented by experts and are free for everyone

You can participate in discussion forums at OSUM and also maintain a blog.

7.3.2 Ohloh.net – Track contributions across projects

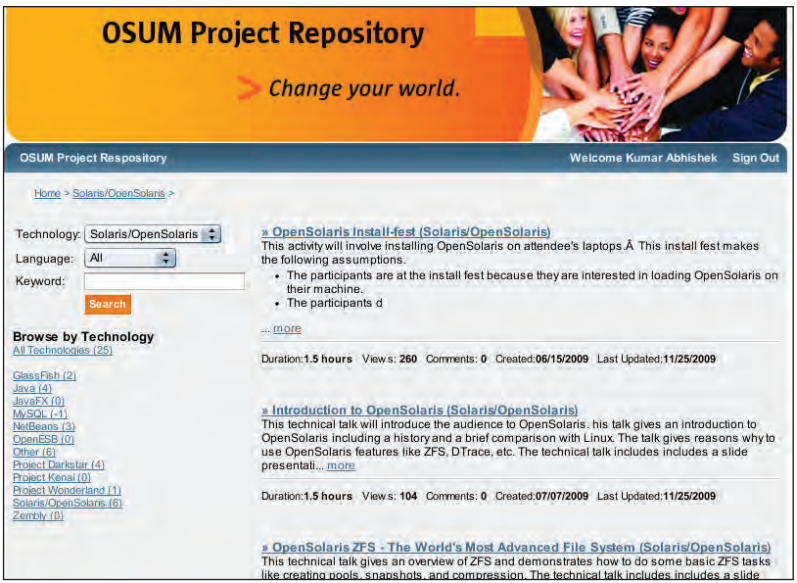


OSUM discussion forums largely sees participation from students all across the globe

Ohloh (http://ohloh.net)is a web site which provides a web services suite and online community platform that aims to map the landscape of open source software development. By retrieving data from revision control repositories (such as CVS, SVN, or Git), Ohloh provides statistics about the longevity of projects, their licenses (including licence conflict information) and software metrics such as source lines of code and commit statistics. The codebase history informs about the amount of activity for each project. Software stacks (list of software applications used by Ohloh's members) and tags are used to calculate the similarity between

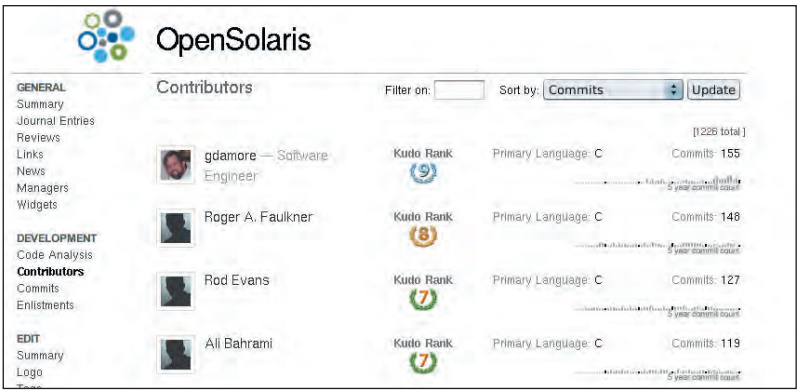
Quick Tip

The most interesting part of OSUM is the OSUM Project Repositories (http://tr.im/osumprojects), which is a list of suggested projects in open source technologies which students can participate in. These projects are selected out of thousands of projects and are not very difficult to work on. It is a good place to look out for interesting topics to work on, if you are trying to identify your first open source project to contribute to.



OpenSolaris Projects listed at OSUM Project Repositories

projects. Contributor statistics are also available, measuring open source developers' experience as observable in code committed to revision control repositories. Social network features (kudos) have been introduced to allow users to rank open source contributors. A KudoRank for each user and open



List of OpenSolaris Contributors who have profiles at Ohloh.net

source contributor on a scale of 1 to 10 is automatically extracted from all kudos in the system.

You can create your own profile at Ohloh and it will reflect your experience in a particular programming language, along with the number of commits you have made into various projects that you are part of.

This can be a very good tool for you to showcase your expertise and experience for rest of the world to see!

Contribute to OpenSolaris

If you are confused where to start working on open source projects, read the Five Step Guide To Contributing Code To OpenSolaris

Conclusion

We are now living in the age of participation and all it takes is an online search for your name to know who you are and what you do. Having a strong presence in the cyberworld is more important than ever. If you make a buzz in the cyberworld with your contributions, you can reap the advantage of it in the real world!

8 Web resources

In OpenSolaris community, there are many places from where you can get tools and help. In this chapter we bring to you some selected places to go out for seeking help or download tools and software. We have also prepared a 5 step guide for those who want to become open source developers and do not know where to start and how to proceed further. And at last, we have a list of things you should do once you finish reading this Fast Track.

OpenSolaris Home

http://www.opensolaris.com

OpenSolaris Community

http://www.opensolaris.org

OpenSolaris How To Guides

http://www.opensolaris.com/learn/howto_guides.html

OpenSolaris Ignite Newsletter

http://tr.im/ignitenewsletter

Software Respositories:

/release

http://pkg.opensolaris.org/release/

/contrib

http://pkg.opensolaris.org/contrib/

/pending

http://jucr.opensolaris.org/pending/

Solaris Documentation

http://docs.sun.com/app/docs/prod/solaris

OpenSolaris Source Code Browser

http://src.opensolaris.org/source/

OpenSolaris Communities List

http://tr.im/oscommunities

8 Web resources

OpenSolaris Planet
http://planet.opensolaris.org/

OpenSolaris Observatory Blogs
http://blogs.sun.com/observatory/

OpenSolaris Developers Forum
http://forums.opensolaris.com/index.jspa

Bangalore OpenSolaris User Group (BOSUG)
http://tr.im/bosug

Mumbai OpenSolaris User Group (MMOSUG)
http://tr.im/bosug

Pune OpenSolaris User Group (POSUG)
http://tr.im/posug

OpenSolaris on Facebook
If you're a Facebook user, you can connect with us at the Indian OpenSolaris User Facebook Group at http://tr.im/osolfacebook

8.1 Five Step Guide to Contribute Code

Step 1
Create an opensolaris account at http://auth.opensolaris.org.

Step 2
Create a SSH public key and update it in your opensolaris.org profile page. To create a SSH key open a terminal and execute following command and upload the ssh.key file which will get created on your desktop.

```
$ ssh-keygen -b 1024 -t dsa -f ~/Desktop/ssh.key
```

You can upload up to 3 SSH public keys to your profile. You will be able to access the code repositories for committing code only from these computers for which you have uploaded the SSH keys.

Grow with OpenSolaris

Now that you are familiar with OpenSolaris, it's time you contributed to this world with your code. Follow the steps and links in this Fast Track to be a part of this rapidly growing community of enthusiastic and passionate OpenSolaris developers.

Step 3

Identify the project on which you would like to work on. Projects in OpenSolaris is sponsored by the concerted community. Here are some areas on which you can work:

Desktop Projects - http://tr.im/desktopprojects
Device Driver Community Projects - http://tr.im/devicedrv
Documentation Projects - http://tr.im/docsproject
Games on OpenSolaris Projects - http://tr.im/gamesproject
Laptop Support on OpenSolaris Projects - http://tr.im/laptopsproject
Software Testing Projects - http://tr.im/testingprojects
Software Porting Projects - http://tr.im/portingprojects

Check out the complete list of communities and projects at http://tr.im/oscommunities.

Step 4

Check the membership criteria for the project of your choice. At this point you should subscribe to the project discussion mailing list. The list of all opensolaris mailing list is available at http://mail.opensolaris.org. Complete the membership requirements and proceed to setting up your development environment.

Step 5

The place where the complete source code of the project is stored is called a code repository. OpenSolaris uses Mercurial code repository software to manage its codebase. Get started with Mercurial Code Repository.

Usage

The Mercurial program is named hg (the chemical symbol for Mercury). Every Mercurial command starts with hg, followed by the command name, followed by any relevant options and arguments. Just running hg on the command line displays a list of commands.

```
$ hg
Mercurial Distributed SCM
```

basic commands (use `hg help` for the full list or option `-v` for details):

`add` the specified files on the next commit
`annotate` show changeset information per file line
`clone` make a copy of an existing repository
`commit` commit the specified files or all outstanding changes
...

The `help` command can be used to get more information about any command.

```
$ hg help diff
hg diff [-a] [-I] [-X] [-r REV1 [-r REV2]] [FILE]...

diff repository (or selected files)
...
```

8.1.1 Getting an initial copy

Use Mercurial's clone subcommand to get a copy of the project repository.

```
$ hg clone  \
ssh://anon@hg.opensolaris.org/hg/onnv/onnv-gate my-copy

requesting all changes
adding changesets
adding manifests
adding file changes
added 3196 changesets with 65633 changes to 42834 files
39543 files updated, 0 files merged, 0 files removed, 0
files unresolved
```

8.1.2 Updating a child repository

Use Mercurial's pull and update subcommands to acquire any changes since your last pull or clone operation. If there are no pending changes, you'll see a result like:

```
$ cd my-copy
$ hg pull
pulling from ssh://anon@hg.opensolaris.org/hg/onnv/
onnv-gate
searching for changes
```

```
no changes found
```

Otherwise, you'll get output informing you of the number of changesets involved:

```
$ hg pull
pulling from ssh://anon@hg.opensolaris.org/hg/onnv/onnv-gate
searching for changes
adding changesets
adding manifests
adding file changes
added 503 changesets with 5608 changes to 4444 files
(run 'hg update' to get a working copy)
$ hg update
4433 files updated, 0 files merged, 376 files removed, 0 files unresolved
```

8.1.3 Using hg bundles of the ON repository

OpenSolaris mercurial repositories are sometimes provided as "bundles" with a .hg extension. There are two types of bundles - Changesets and Incremental. Changeset bundles are complete standalone bundles. To get a working copy of this bundle, download it from the opensolaris.org downloads page and do the following steps.

```
cd /export/work
hg init
hg unbundle {path}/on-hg-bundle-YYYYMMDD.hg
hg update
```

Incremental bundles are typically named as "on-hg-bundle_yyyymmdd-YYYYMMDD.hg" and contain only incremental changes. They can only be applied to a working copy which contains all changesets up to yyyymmdd and no more. This is verified before the unbundling process by checking the parent property (hg parent) of the bundle. You can apply an incremental bundle to a working copy using the following commands:

```
cd /export/work
hg unbundle on-hg-bundle_yyyymmdd-YYYYMMDD.hg
```

```
hg update
```

Once you have a working copy, you can clone it and do other operations described below.

8.1.4 Editing files

Mercurial automatically keeps track of modified files.

Adding files

To add files to the workspace use the add command.

```
$ hg add file1.c ...
```

These files will be added to the repository on the next commit.

Examining differences

Use the `diff` command to examine differences.

```
$ hg diff
```

Without any arguments, `hg diff` will print out the differences between all modified files in the repository. Differences between files are shown using the unified diff format.

Commiting the changes

The `commit` command commits the changes.

```
$ hg commit
```

This command drops us into an editor (configured using the EDITOR and HGEDITOR environment variables) where we can add the putback comments. Once we save the changes and quit the editor, the changes are committed to the repository.

Putting back the changes to the parent repository

All changes up to this point have been in the working clone repository. The changes can be put back to the parent repository using the push command.

```
$ hg push /ws/onnv-gate
```

Read the full tutorial on how to manage Mercurial Repositories at http://hub.opensolaris.org/bin/view/Community+Group+tools/hg_help.

Once you make a code putback to the parent repository, you make a contribution. Congratulations, you just became a open source contributor!

8.2 Checklists

Here are some action points for you to act on, based on how you want to use OpenSolaris.

8.2.1 User Checklist

- Install OpenSolaris. Use LiveCD bundled with Digit or download from http://www.opensolaris.com
- Configure /contrib and /extra repository in Package Manager. **Check out chapter 3 on how to do so.**
- Register your copy of OpenSolaris and Create an account at http://opensolaris.org
- Join a OpenSolaris User Group mailing list. If you are a new user of OpenSolaris we recommend you subscribe to Mumbai OSUG (http://tr.im/mumbaiosug)
- If you have used OpenSolaris in past, you can join Bangalore OSUG, which is a usergroup of advanced OpenSolaris users (http://tr.im/bosug)
- Join the Indian OpenSolaris Users Facebook Group http://tr.im/osolfacebook

8.2.2 Developer Checklist

- Install OpenSolaris. Use LiveCD bundled with Digit or download from http://www.opensolaris.com
- Install Sun Studio using Package Manager
- Install Netbeans using Package Manager
- Install gcc-dev using Package Manager
- Add /dev repository to your Package Manager
- Create an opensolaris account at http://opensolaris.org
- Update your SSH keys in your OpenSolaris profile
- Identify your project.

The list of all projects is at http://tr.im/oscommunities

Win prizes!

Now that you have learned so much about open source and OpenSolaris, go ahead and test your skills in the OpenSolaris Geek Hunt. For more details, go to http://www.thinkdigit.com/opensolaris. If you're the geek we're looking for, you could win yourself an Acer Netbook and more!

8 Web resources

- Join the mailing list your project
- Join the User Groups and Facebook group at http://tr.im/osolfacebook

9 The future

What's new in OpenSolaris 2010.03?

OpenSolaris 2010.03 will be released next month in March, which will be the next major release after OpenSolaris 2009.06. These are some of the new features which will be added to OpenSolaris 2010.03.

Facet support in IPS - Facet support in IPS allows users to filter out what they choose to install onto their system, for example, a particular language or locale choice, developer heads or documentation. Users and administrators can now choose to install particular parts of a software package onto their system, without the complexity of having to install a myriad of individual software packages supporting that choice. The addition of facet support allows ISVs to package their application as a single software package, and provide a set of pre-defined filters.

VM Constructor - The VM Constructors allows users to create virtual images based on selections of software, exported as an OVF appliance. Virtualization provides a popular and convenient way of deploying services within an enterprise environment. The VM constructor allows users to create specific virtualized images to be used in any hypervisor that supports the OVF 1.0 specification.

Extended partition support and partition editor: OpenSolaris now supports the ability to install and boot from a logical drive within an extended partition. The gparted tool on the OpenSolaris LiveCD allows the ability to resize/reconfigure existing partitions to more easily install OpenSolaris. OpenSolaris can now be installed into an extended partition, allowing users the additional flexibility with other operating systems in a multi-boot environment. The user can now easily modify existings partitions from the LiveCD environment with a simple to use partition editor.

ZFS deduplication and ZFS user/group quotas: ZFS deduplication is the process of eliminating duplicate copies of data. Data quotas can now be set for users and groups. ZFS deduplication can be used to preserve space on disk when multiple snapshot or file system clones have been made (including boot environments). ZFS user and group quotas provide missing functionality allowing administrators easy control over disk usage when managing servers with thousands of users.

Addition of Visual Panels in OpenSolaris default installation: Although

previously available in the software repository, Visual Panels (user and groups, firewall, system shares, kernel dumps) has been added to the default OpenSolaris installation. Visual panels provides administrators a familiar graphical interface for configuring certain aspects of a system.

Boomer Sound System: Boomer, based on OSS, is a new sound system available on OpenSolaris. The introduction of Boomer provides a much improved audio experience to OpenSolaris, based on OSS (Open Sound System), providing greater support for advanced audio hardware (5.1, 7.1, high definition audio) and a improved programming interface support for the latest audio applications.

CUPS: CUPS will be now the default print system on OpenSolaris. The introduction of CUPS as the default print system is another step towards familiarization, predominantly with Linux based platforms where CUPS is a popular and easy to use printing system.

Updating OpenSolaris 2009.06 to OpenSolaris 2010.03 is a very simple task! Go ahead and install OpenSolaris 2009.06 and when OpenSolaris 2010.03 will be released, your installation will be automatically updated by using the Update Manager.

RNI REGN. NO. MAH ENG/2001/3791

Fast track to OpenSolaris